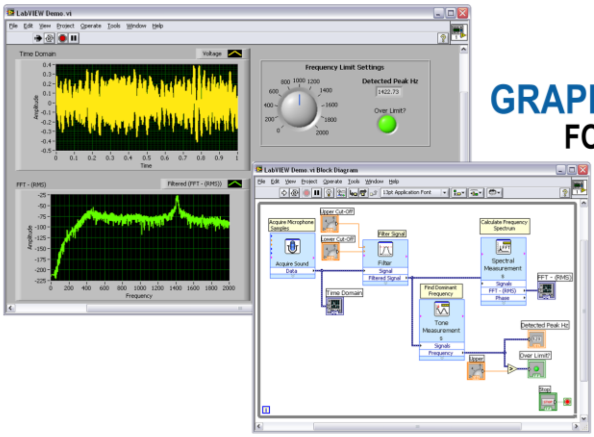
Introduction to LabVIEW



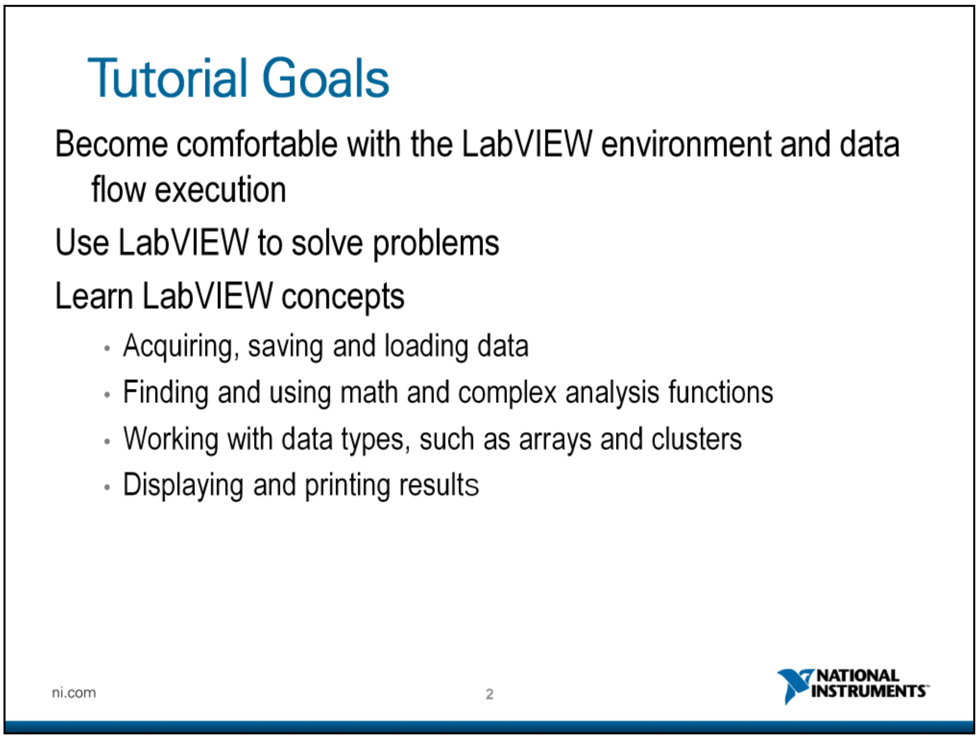
GRAPHICAL PROGRAMMING
FOR ENGINEERS AND SCIENTISTS

NATIONAL INSTRUMENTS LabVIEW™

3-Hour Hands-On Tutorial

ni.com

NATIONAL INSTRUMENTS™



Tutorial Goals

Become comfortable with the LabVIEW environment and data flow execution

Use LabVIEW to solve problems

Learn LabVIEW concepts

- Acquiring, saving and loading data
- Finding and using math and complex analysis functions
- Working with data types, such as arrays and clusters
- Displaying and printing results

ni.com 2 NATIONAL INSTRUMENTS

This course prepares you to do the following:

- Use LabVIEW to create applications.
- Understand front panels, block diagrams, and icons and connector panes.
- Use built-in LabVIEW functions.
- Create and save programs in LabVIEW so you can use them as subroutines.
- Create applications that use plug-in DAQ devices.

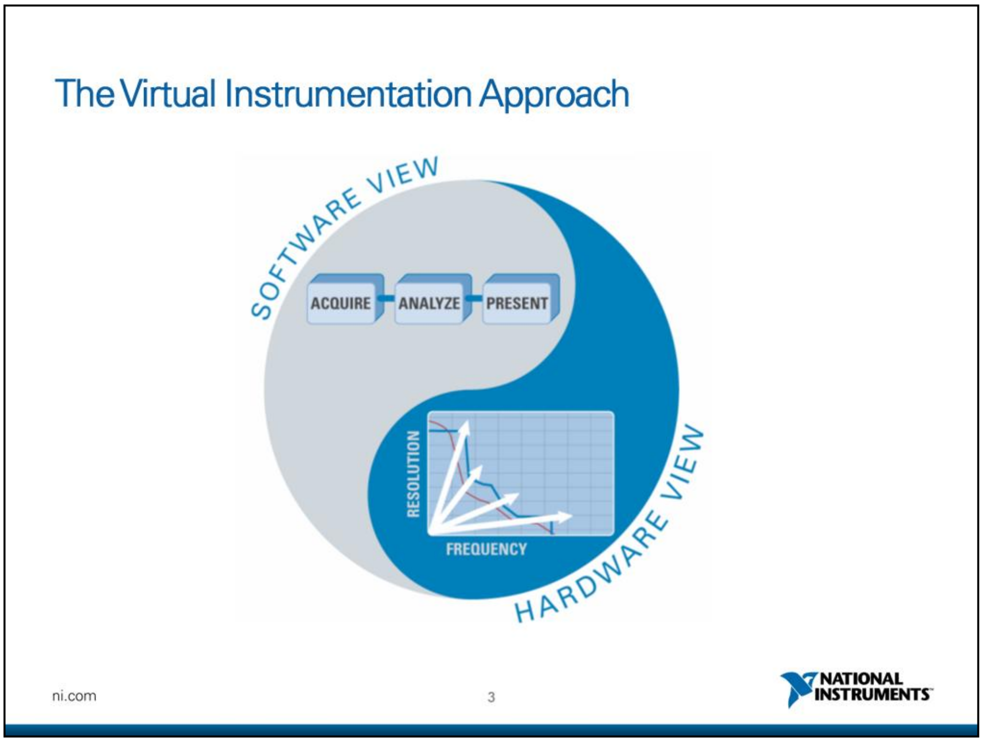
This course does *not* describe any of the following:

- Programming theory
- Every built-in LabVIEW function or object
- Analog-to-digital (A/D) theory

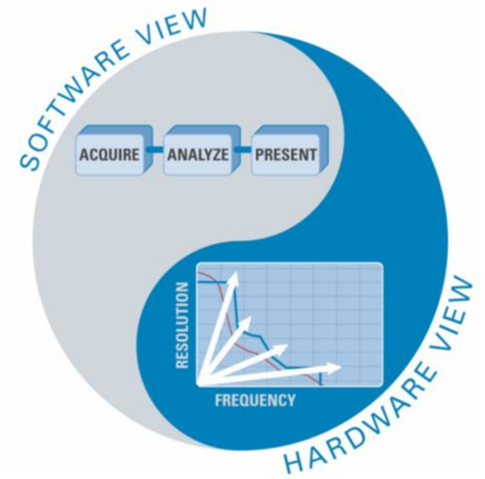
NI does provide free reference materials on the above topics on `ni.com`.

The *LabVIEW Help* is also very useful:

LabVIEW»Help»Search the LabVIEW Help…



The Virtual Instrumentation Approach

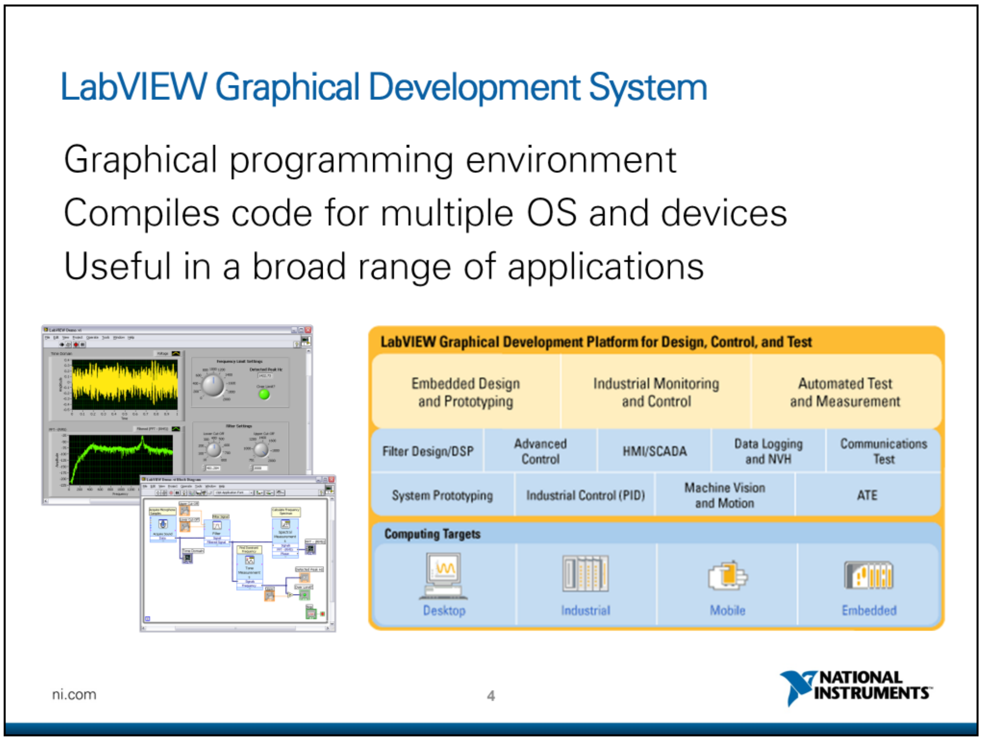


ni.com 3

Virtual Instrumentation

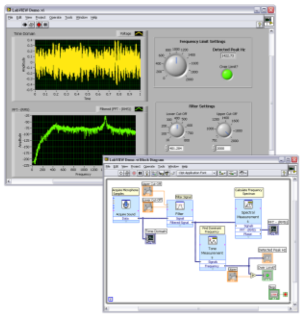
For more than 30 years, National Instruments has revolutionized the way engineers and scientists in industry, government, and academia approach measurement and automation. Leveraging PCs and commercial technologies, virtual instrumentation increases productivity and lowers costs for test, control, and design applications through easy-to-integrate software, such as NI LabVIEW, and modular measurement and control hardware for PXI, PCI, USB, and Ethernet.

With virtual instrumentation, engineers use graphical programming software to create user-defined solutions that meet their specific needs, which is a great alternative to proprietary, fixed-functionality traditional instruments. Additionally, virtual instrumentation capitalizes on the ever-increasing performance of personal computers. For example, in test, measurement, and control, engineers have used virtual instrumentation to downsize automated test equipment (ATE) while experiencing up to a 10 times increase in productivity gains at a fraction of the cost of traditional instrument solutions.



LabVIEW Graphical Development System

Graphical programming environment
Compiles code for multiple OS and devices
Useful in a broad range of applications



LabVIEW Graphical Development Platform for Design, Control, and Test

Embedded Design and Prototyping	Industrial Monitoring and Control	Automated Test and Measurement

Filter Design/DSP	Advanced Control	HMI/SCADA	Data Logging and NVH	Communications Test

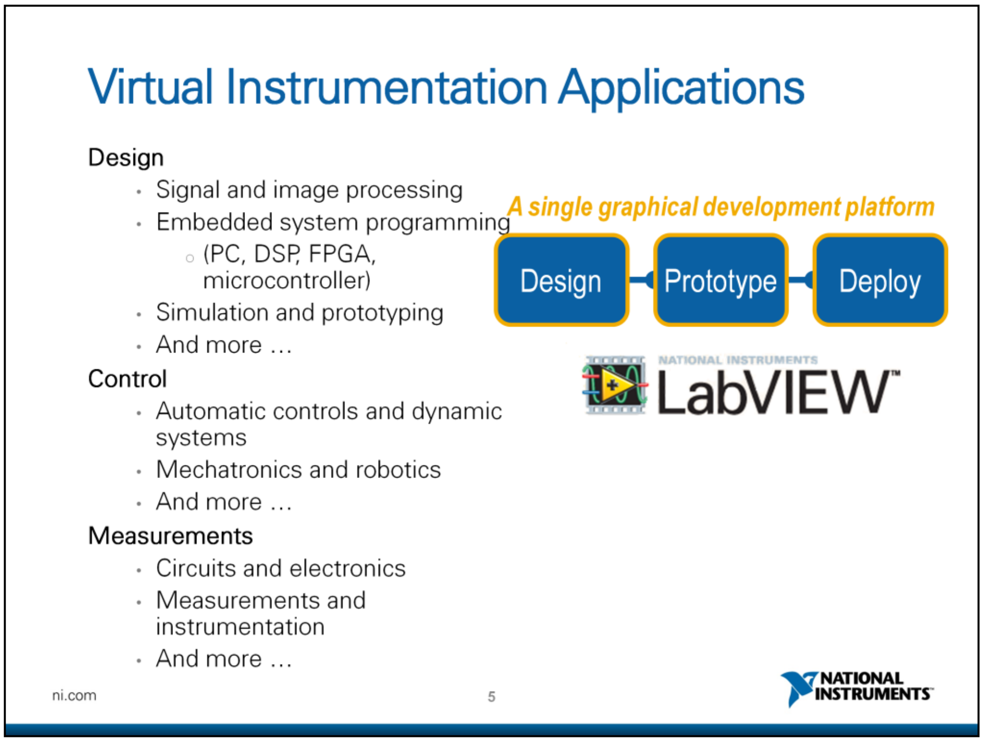
System Prototyping	Industrial Control (PID)	Machine Vision and Motion	ATE

Computing Targets

Desktop	Industrial	Mobile	Embedded

ni.com 4 NATIONAL INSTRUMENTS

National Instruments LabVIEW is an industry-leading software tool for designing test, measurement, and control systems. Since its introduction in 1986, engineers and scientists worldwide who have relied on NI LabVIEW graphical development for projects throughout the product design cycle have gained improved quality, shorter time to market, and greater engineering and manufacturing efficiency. By using the integrated LabVIEW environment to interface with real-world signals, analyze data for meaningful information, and share results, you can boost productivity throughout your organization. Because LabVIEW has the flexibility of a programming language combined with built-in tools designed specifically for test, measurement, and control, you can create applications that range from simple temperature monitoring to sophisticated simulation and control systems. No matter what your project is, LabVIEW has the necessary tools to make you successful quickly.



Virtual Instrumentation Applications

Design
- Signal and image processing
- Embedded system programming
  - (PC, DSP, FPGA, microcontroller)
- Simulation and prototyping
- And more …

Control
- Automatic controls and dynamic systems
- Mechatronics and robotics
- And more …

Measurements
- Circuits and electronics
- Measurements and instrumentation
- And more …

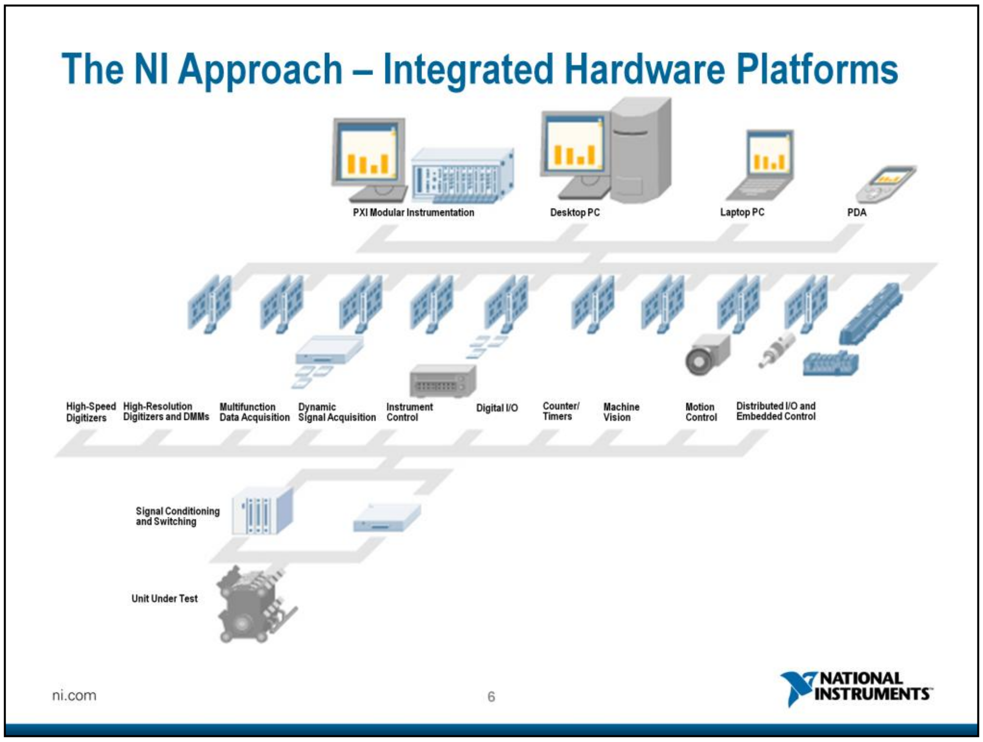
A single graphical development platform

ni.com 5 NATIONAL INSTRUMENTS

Virtual Instrumentation Applications

Virtual instrumentation is effective in many different types of applications, from design to prototyping to deployment. The LabVIEW platform provides specific tools and models to meet specific application challenges, ranging from designing signal processing algorithms to making voltage measurements, and can target any number of platforms from the desktop to embedded devices – with an intuitive, powerful graphical paradigm.

LabVIEW scales from design and development on PCs to several embedded targets, from rugged toaster-size prototypes to embedded systems on chips. **LabVIEW streamlines system design with a single graphical development platform**. In doing so, it encompasses better management of distributed, networked systems because as the targets for LabVIEW grow varied and embedded, you need to be able to more easily distribute and communicate between the various LabVIEW code pieces in your system.



The NI Approach – Integrated Hardware Platforms

PXI Modular Instrumentation
Desktop PC
Laptop PC
PDA
High-Speed Digitizers
High-Resolution Digitizers and DMMs
Multifunction Data Acquisition
Dynamic Signal Acquisition
Instrument Control
Digital I/O
Counter/ Timers
Machine Vision
Motion Control
Distributed I/O and Embedded Control
Signal Conditioning and Switching
Unit Under Test

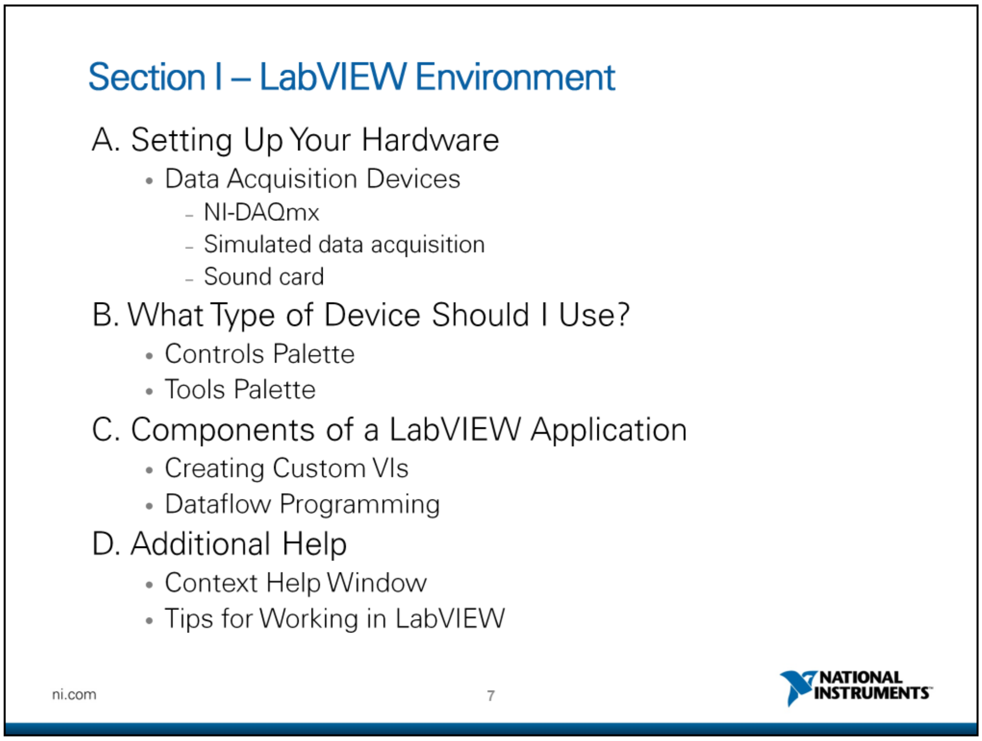
ni.com 6

Integrated Hardware Platforms

A virtual instrument consists of an industry-standard computer or workstation equipped with powerful application software, cost-effective hardware such as plug-in boards, and driver software, which together perform the functions of traditional instruments.

Virtual instruments represent a fundamental shift from traditional hardware-centered instrumentation systems to software-centered systems that exploit the computing power, productivity, display, and connectivity capabilities of popular desktop computers and workstations.

Although the PC and integrated circuit technology have experienced significant advances in the last two decades, software truly offers the flexibility to build on this powerful hardware foundation to create virtual instruments, providing better ways to innovate and significantly reduce cost. With virtual instruments, engineers and scientists build measurement and automation systems that suit their needs exactly (user-defined) instead of being limited by traditional fixed-function instruments (vendor-defined).



Section I – LabVIEW Environment

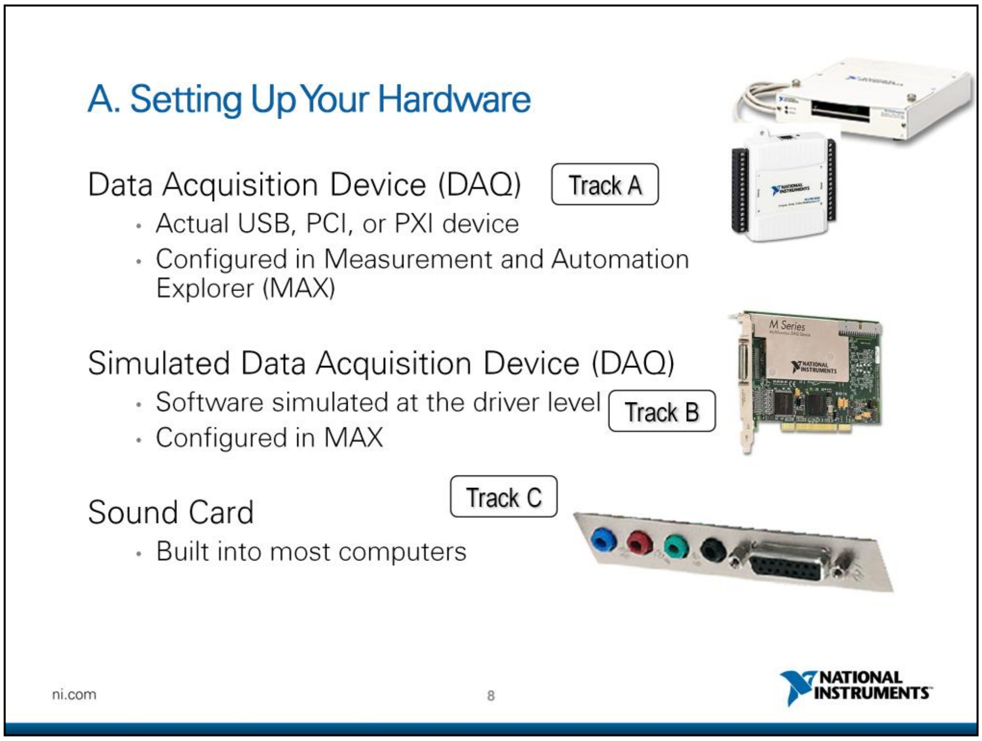
A. Setting Up Your Hardware
- Data Acquisition Devices
  - NI-DAQmx
  - Simulated data acquisition
  - Sound card

B. What Type of Device Should I Use?
- Controls Palette
- Tools Palette

C. Components of a LabVIEW Application
- Creating Custom VIs
- Dataflow Programming

D. Additional Help
- Context Help Window
- Tips for Working in LabVIEW

NATIONAL INSTRUMENTS



A. Setting Up Your Hardware

Data Acquisition Device (DAQ) Track A

- Actual USB, PCI, or PXI device
- Configured in Measurement and Automation Explorer (MAX)

Simulated Data Acquisition Device (DAQ)

- Software simulated at the driver level Track B
- Configured in MAX

Track C

Sound Card

- Built into most computers

ni.com 8

This LabVIEW course is designed for audiences with or without access to National Instruments hardware.

Each exercise is divided into three tracks, A, B, and C

Track A is designed to be used with hardware supported by the NI-DAQmx driver. This includes most USB, PCI, and PXI data acquisition devices with analog input. Some signal conditioning and excitation (external power) is required to use a microphone with a DAQ device.

Track B is designed to be used with no hardware. You can simulate the hardware with NI-DAQmx Version 8.0 and later. This is done by using the NI-DAQmx Simulated Device option in the Create New menu of MAX. The simulated device's driver is loaded, and programs using it are fully verified.

Track C is designed to be used with a standard sound card and microphone. LabVIEW includes simple VIs for analog input and analog output using the sound card built into many PCs. This is convenient for laptops because the sound card and microphone are usually already built-in.

Setting Up Your Hardware for Your Selected Track

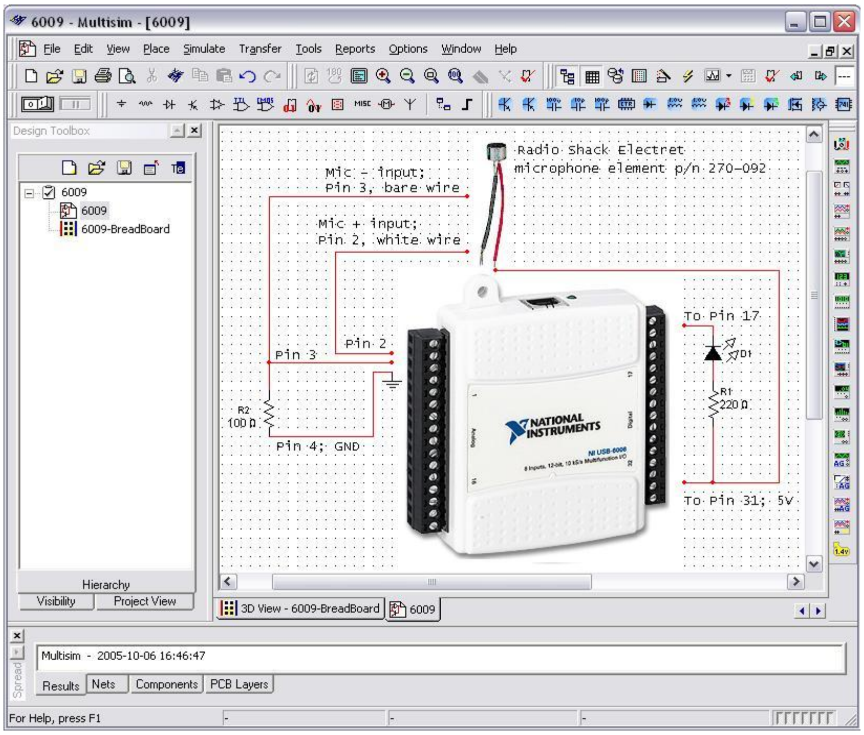
Track A – NI Data Acquisition with Microphone

USB-6009, Microphone and LED

Suggested Hardware

Qty	Part Number	Description	Supplier
1	779321-22	Low-Cost USB DAQ	National Instruments
1	270-092	Electret Microphone	RadioShack
1		100 Ω Resistor	RadioShack
1		220 Ω Resistor	RadioShack
1	276-307	Light Emitting Diode (LED)	RadioShack

The following schematic was drawn with NI **Multisim**, a widely used SPICE schematic capture and simulation tool. Visit `ni.com/Multisim` for more info.



Track B – Simulated NI Data Acquisition

NI-DAQmx Software Version 8.0 or later

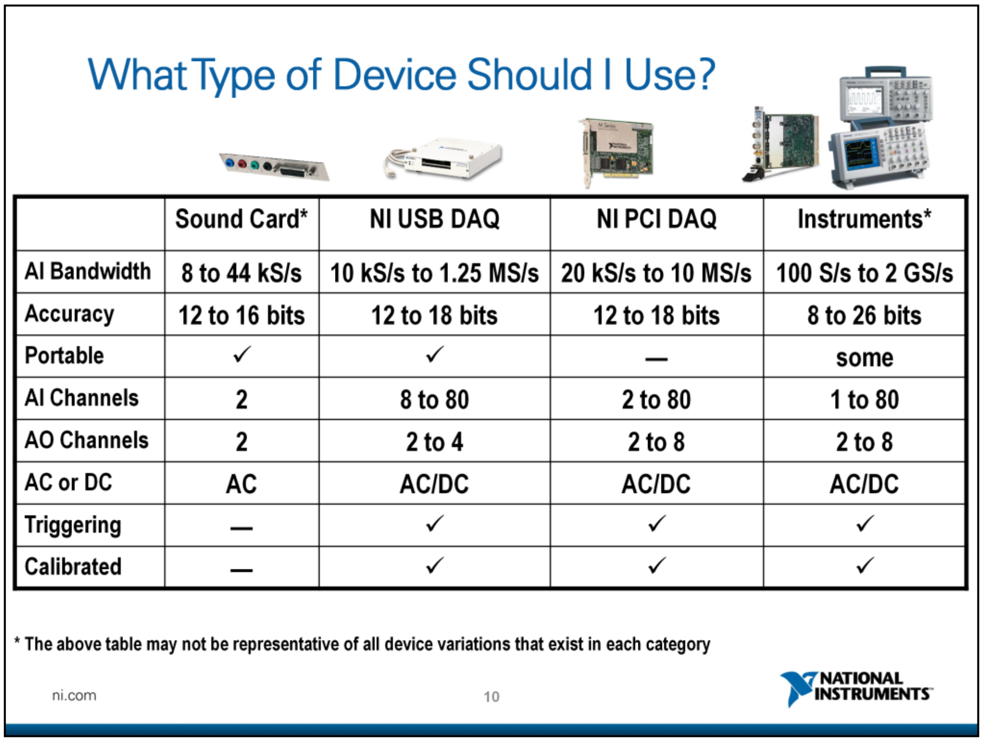
Track C – Third-Party Sound Card

Sound card and Microphone

Suggested Hardware

Qty	Part Number	Description	Supplier
1		Standard Plug-In PC Microphone*	RadioShack

* Laptops often have a built-in microphone (no plug-in microphone is required)



What Type of Device Should I Use?

	Sound Card*	NI USB DAQ	NI PCI DAQ	Instruments*
AI Bandwidth	8 to 44 kS/s	10 kS/s to 1.25 MS/s	20 kS/s to 10 MS/s	100 S/s to 2 GS/s
Accuracy	12 to 16 bits	12 to 18 bits	12 to 18 bits	8 to 26 bits
Portable	✓	✓	—	some
AI Channels	2	8 to 80	2 to 80	1 to 80
AO Channels	2	2 to 4	2 to 8	2 to 8
AC or DC	AC	AC/DC	AC/DC	AC/DC
Triggering	—	✓	✓	✓
Calibrated	—	✓	✓	✓

* The above table may not be representative of all device variations that exist in each category

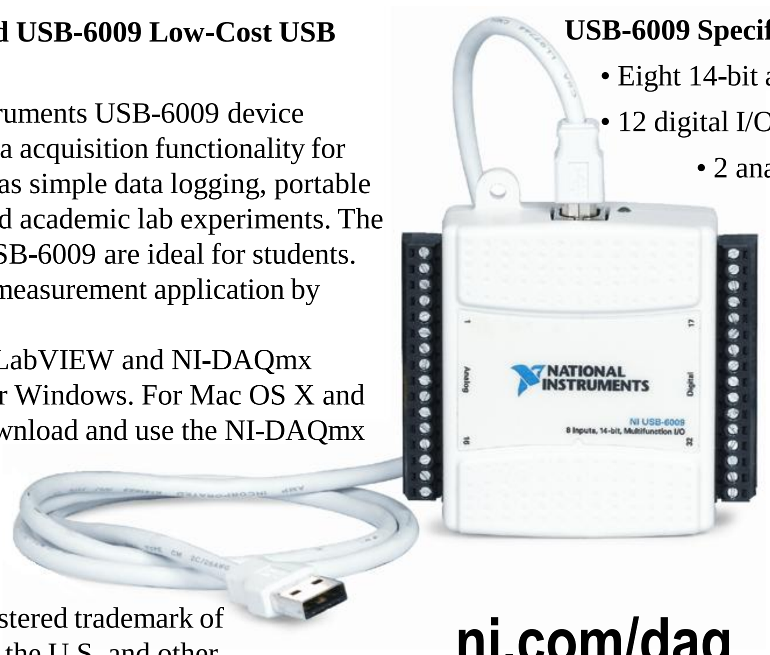
ni.com 10

What type of device should I use?

There are many types of data acquisition and control devices on the market. A few have been highlighted above. The trade-off usually falls between sampling rate (samples/second), resolution (bits), number of channels, and data transfer rate (usually limited by “bus” type: USB, PCI, PXI, and so on). Multifunction DAQ (data acquisition) devices are ideal because they can be used in a wide range of applications.

NI USB-6008 and USB-6009 Low-Cost USB DAQ

The National Instruments USB-6009 device provides basic data acquisition functionality for applications such as simple data logging, portable measurements, and academic lab experiments. The USB-6008 and USB-6009 are ideal for students. Create your own measurement application by programming the
USB-6009 using LabVIEW and NI-DAQmx driver software for Windows. For Mac OS X and Linux® users, download and use the NI-DAQmx Base driver.

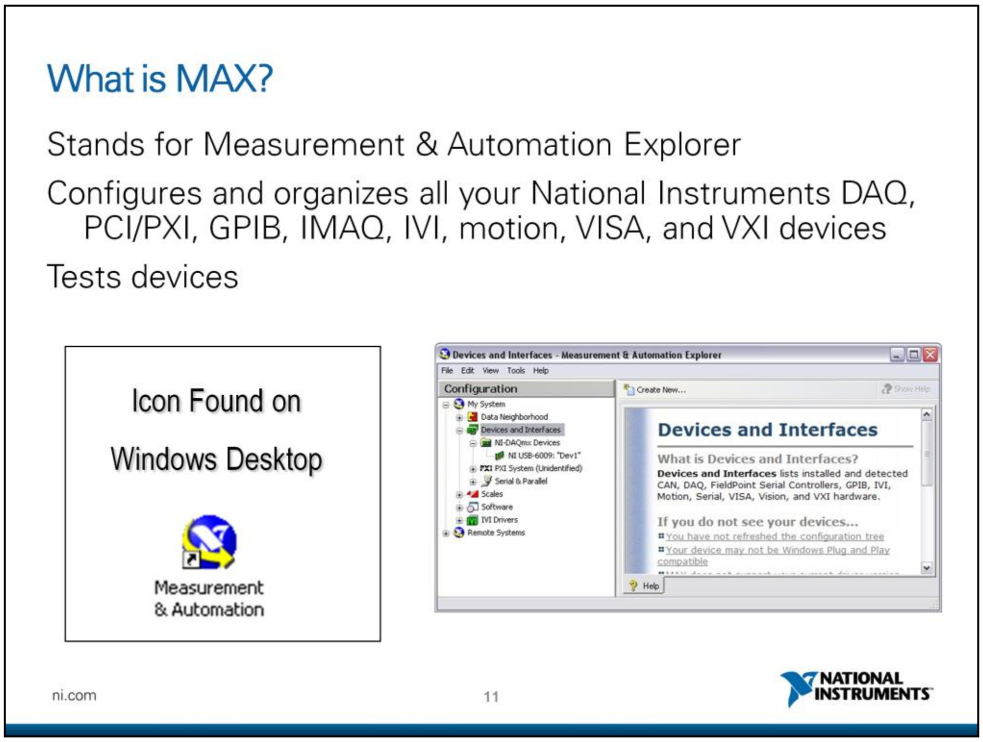


USB-6009 Specifications:

- Eight 14-bit analog inputs
- 12 digital I/O lines
- 2 analog outputs
- 1 counter

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

ni.com/daq



What is MAX?

Stands for Measurement & Automation Explorer

Configures and organizes all your National Instruments DAQ, PCI/PXI, GPIB, IMAQ, IVI, motion, VISA, and VXI devices

Tests devices

Icon Found on Windows Desktop

Devices and Interfaces

What is Devices and Interfaces?

Devices and Interfaces lists installed and detected CAN, DAQ, FieldPoint Serial Controllers, GPIB, IVI, Motion, Serial, VISA, Vision, and VXI hardware.

If you do not see your devices...

- You have not refreshed the configuration tree
- Your device may not be Windows Plug and Play compatible

ni.com 11 NATIONAL INSTRUMENTS

The next level of software is called Measurement & Automation Explorer (MAX). MAX is a software interface that gives you access to all of your National Instruments DAQ, GPIB, IMAQ, IVI, Motion, VISA, and VXI devices. The shortcut to MAX appears on your desktop after installation. A picture of the icon is shown above. MAX is mainly used to configure and test your National Instruments hardware, but it does offer other functionality such as checking to see if you have the latest version of NI-DAQmx installed. When you run an application using NI-DAQmx, the software reads the MAX configuration to determine the devices you have configured. Therefore, you must configure DAQ devices first with MAX.

The functionality of MAX falls into six categories:

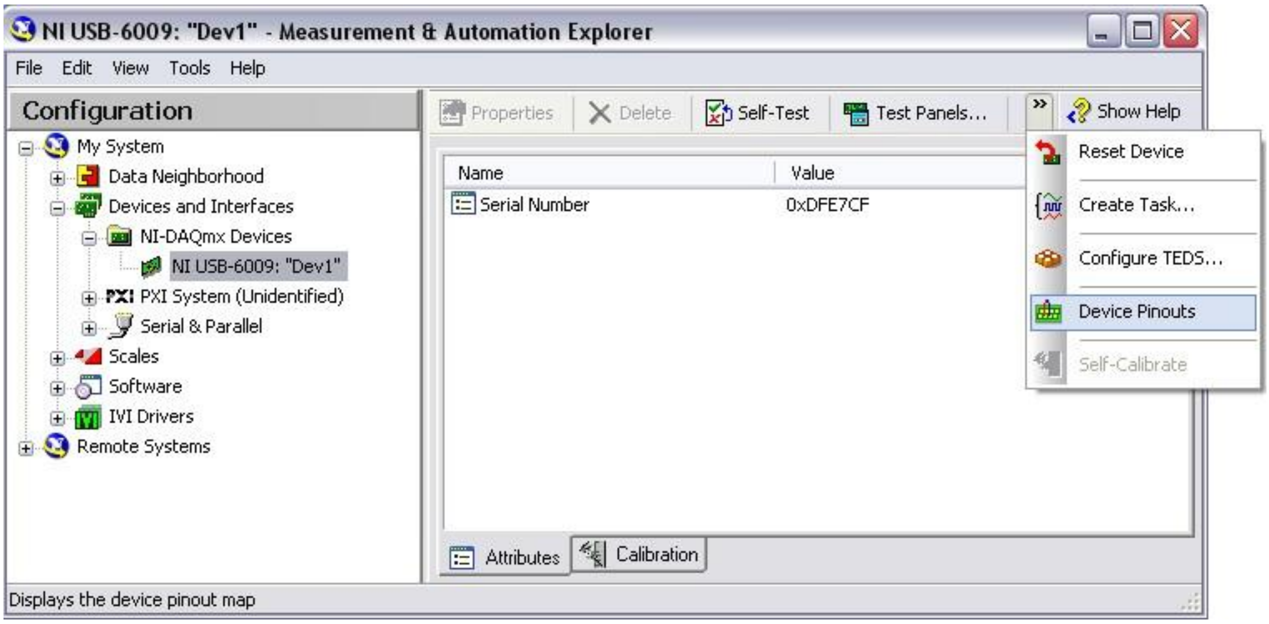
- Data Neighborhood
- Devices and Interfaces
- IVI Instruments
- Scales
- Historical Data
- Software

This course will focus on Data Neighborhood, Devices and Interfaces, Scales, Software, and you will learn about the functionality each one offers.

Exercise 1 – Testing Your Device (Track A)

In this exercise, use Measurement and Automation Explorer (MAX) to test your USB-6009 DAQ device.

1. Launch MAX by double-clicking the icon on the desktop or by selecting **Start» Programs»National Instruments»Measurement & Automation**.
2. Expand the **Devices and Interfaces** section to view the installed National Instruments devices. MAX displays the National Instruments hardware and software in the computer.
3. Expand the **NI-DAQmx Devices** section to view the installed hardware that is compatible with NI-DAQmx. The device number appears in quotes following the device name. The data acquisition VIs use this device number to determine which device performs DAQ operations. Your hardware is listed as NI USB-6009: “Dev1.”
4. Perform a self-test on the device by right-clicking it in the configuration tree and choosing **Self-Test** or clicking **Self-Test** along the top of the window. This tests the system resources assigned to the device. The device should pass the test because it is already configured.
5. Check the pinout for your device. Right-click the device in the configuration tree and select **Device Pinouts** or click **Device Pinouts** along the top of the center window.
6. Open the test panels. Right-click the device in the configuration tree and select **Test Panels…** or click **Test Panels…** along the top of the center window. With test panels, you can test the available functionality of your device, analog input/output, digital input/output, and counter input/output without doing any programming.
7. On the **Analog Input** tab of the test panels, change **Mode** to “Continuous” and **Rate** to “10,000 Hz.” Click **Start** and hum or whistle into your microphone to observe the signal that is plotted. Click **Stop** when you are done.
8. On the **Digital I/O** tab, notice that initially the port is configured to be all input. Observe under **Select State** the LEDs that represent the state of the input lines. Click the **All Output** button under **Select Direction**. Notice you now have switches under **Select State** to specify the output state of the different lines. Toggle line 0 and watch the LED light up. Click **Close** to close the test panels.
9. Close MAX.



Name	Value
Serial Number	0xDFE7CF

Displays the device pinout map

End of Exercise 1 (Track A)

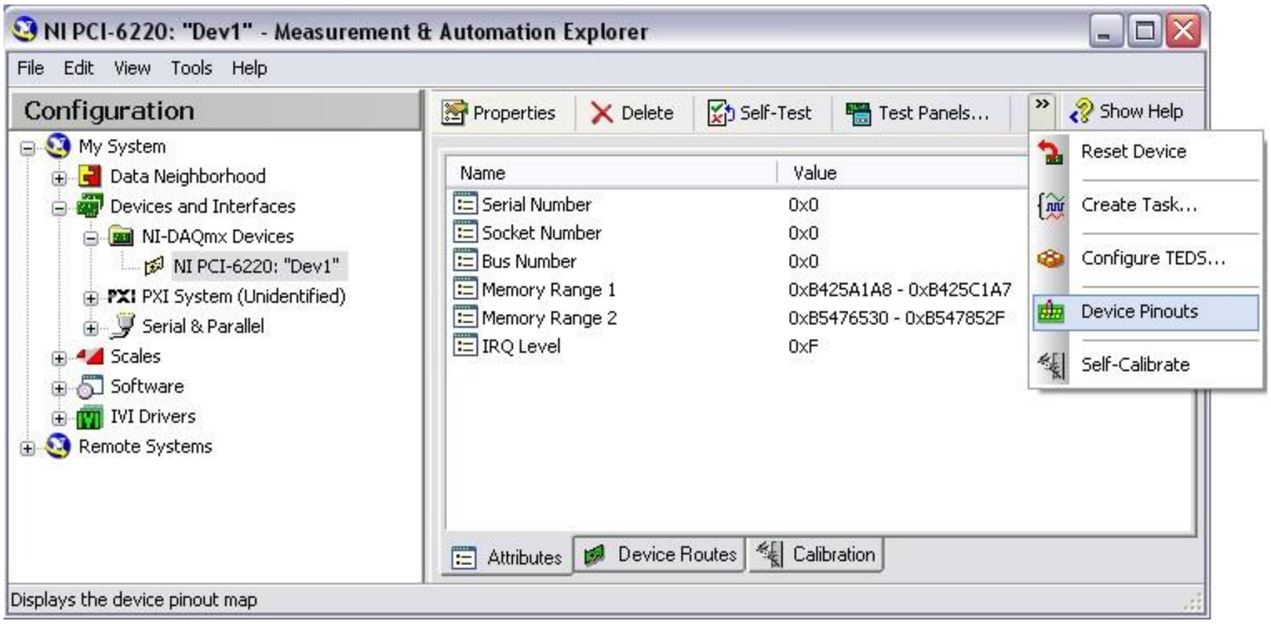
Exercise 1 – Setting Up Your Device (Track B)

In this exercise, use Measurement and Automation Explorer (MAX) to configure a simulated DAQ device.

1. Launch MAX by double-clicking the icon on the desktop or by selecting **Start» Programs»National Instruments»Measurement & Automation**.
2. Expand the **Devices and Interfaces** section to view the installed National Instruments devices. MAX displays the National Instruments hardware and software in the computer. The device number appears in quotes following the device name. The data acquisition VIs use this device number to determine which device performs DAQ operations.
3. Create a simulated DAQ device for use later in this course. Simulated devices are a powerful tool for development without having hardware physically installed in your computer. Right-click **Devices and Interfaces** and select **Create New…» NI-DAQmx Simulated Device**. Click **Finish**. The simulated device appears yellow in color.
4. Expand the M Series DAQ section. Select **PCI-6220** or any other PCI device of your choice. Click **OK**.
5. The NI-DAQmx Devices folder expands to show a new entry for PCI-6220: “Dev1.” You have now created a simulated device.
6. Perform a self-test on the device by right-clicking it in the configuration tree and choosing **Self-Test** or clicking **Self-Test** along the top of the window. This tests the system resources assigned to the device. The device should pass the test because it is already configured.
7. Check the pinout for your device. Right-click the device in the configuration tree and select **Device Pinouts** or click **Device Pinouts** along the top of the center window.
8. Open the test panels. Right-click the device in the configuration tree and select **Test Panels…** or click **Test Panels…** along the top of the center window. The test panels allow you to test the available functionality of your device, analog input/output, digital input/output, and counter input/output without doing any programming.
9. On the **Analog Input** tab of the test panels, change **Mode** to “Continuous.” Click **Start** and observe the signal that is plotted. Click **Stop** when you are done.

10. On the **Digital I/O** tab, notice that initially the port is configured to be all input. Observe under **Select State** the LEDs that represent the state of the input lines. Click the **All Output** button under **Select Direction**. Note that you now have switches under **Select State** to specify the output state of the different lines. Click **Close** to close the test panels.

11. Close MAX.



Name	Value
Serial Number	0x0
Socket Number	0x0
Bus Number	0x0
Memory Range 1	0xB425A1A8 - 0xB425C1A7
Memory Range 2	0xB5476530 - 0xB547852F
IRQ Level	0xF

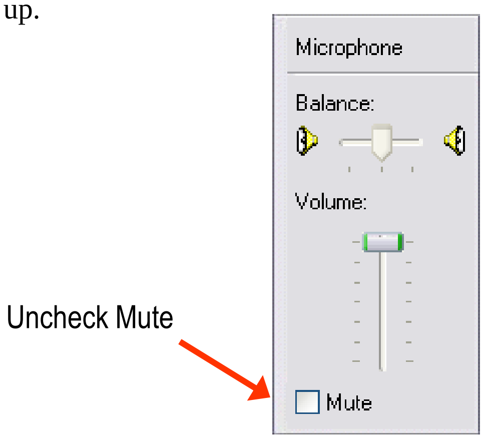
Displays the device pinout map

End of Exercise 1 (Track B)

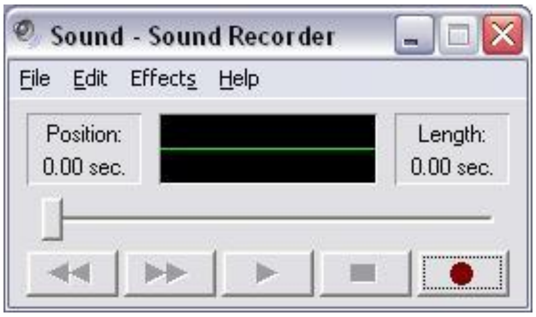
Exercise 1 – Setting Up Your Device (Track C)

In this exercise, use Windows utilities to verify your sound card and prepare it for use with a microphone.

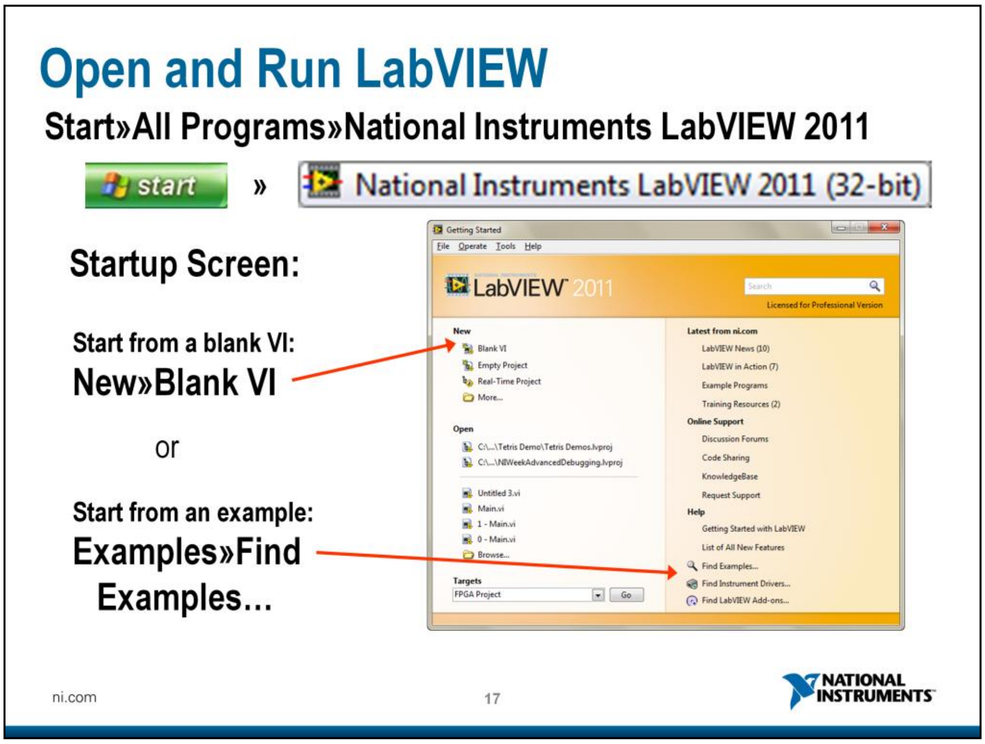
1. Prepare your microphone for use. Double-click the volume control icon on the task bar to open up the configuration window. You also can find the sound configuration window from the Windows Control Panel: **Start»Settings»Control Panel»Sounds and Audio Devices»Advanced**.
2. If you do not see a microphone section, go to **Options»Properties**»**Recording** and place a checkmark in the box next to **Microphone**. This displays the microphone volume control. Click **OK**.
3. Uncheck the **Mute** box if it is not already unchecked. Make sure that the volume is turned up.



4. Close the volume control configuration window.
5. Open the sound recorder by selecting **Start**»**Programs»Accessories» Entertainment»Sound Recorder**.
6. Click the record button and speak into your microphone. Notice how the sound signal is displayed in the sound recorder.
7. Click **Stop** and close the sound recorder without saving changes when you are finished.



End of Exercise 1 (Track C)



Open and Run LabVIEW

Start»All Programs»National Instruments LabVIEW 2011

start » National Instruments LabVIEW 2011 (32-bit)

Startup Screen:

Start from a blank VI:
New»Blank VI

or

Start from an example:
Examples»Find Examples…

ni.com 17 NATIONAL INSTRUMENTS

LabVIEW

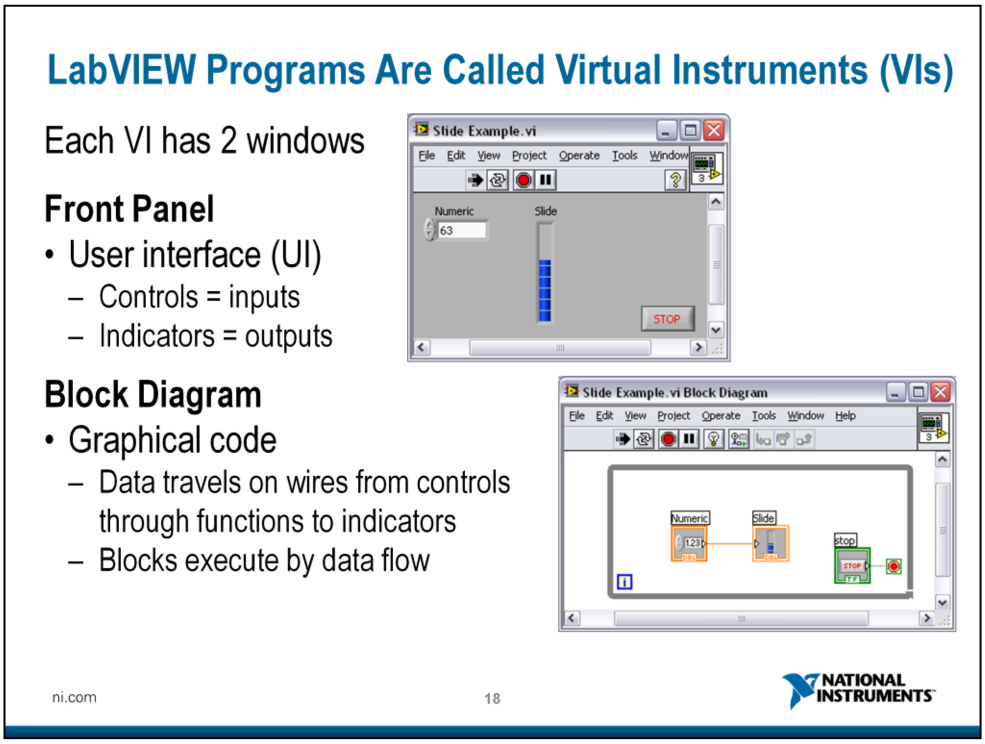
LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution order.

You can purchase several add-on software toolkits for developing specialized applications. All the toolkits integrate seamlessly in LabVIEW. Refer to the National Instruments Web site for more information about these toolkits.

LabVIEW also includes several wizards to help you quickly configure your DAQ devices and computer-based instruments and build applications.

LabVIEW Example Finder

LabVIEW features hundreds of example VIs you can use and incorporate into VIs that you create. In addition to the example VIs that are shipped with LabVIEW, you can access hundreds of example VIs on the NI Developer Zone (`zone.ni.com`). You can modify an example VI to fit an application, or you can copy and paste from one or more examples into a VI that you create.



LabVIEW Programs Are Called Virtual Instruments (VIs)

Each VI has 2 windows

Front Panel
- User interface (UI)
  - Controls = inputs
  - Indicators = outputs

Block Diagram
- Graphical code
  - Data travels on wires from controls through functions to indicators
  - Blocks execute by data flow

Slide Example.vi
File Edit View Project Operate Tools Window
Numeric
63
Slide
STOP
Slide Example.vi Block Diagram
File Edit View Project Operate Tools Window Help
Numeric
123
Slide
stop
STOP
TF

ni.com 18 NATIONAL INSTRUMENTS

LabVIEW programs are called virtual instruments (VIs).

Controls are inputs and indicators are outputs.

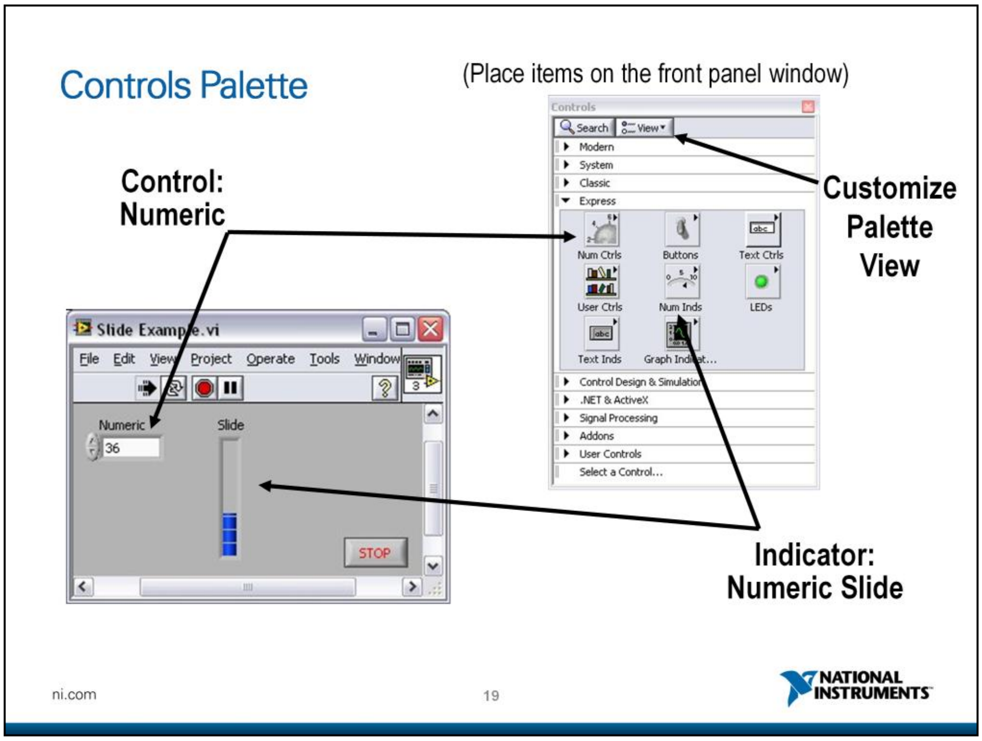
Each VI contains three main parts:

- Front panel – How the user interacts with the VI
- Block diagram – The code that controls the program
- Icon/connector – The means of connecting a VI to other VIs

In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel. You then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

You interact with the front panel when the program is running. You can control the program, change inputs, and see data updated in real time. Controls are used for inputs such as adjusting a slide control to set an alarm value, turning a switch on or off, or stopping a program. Indicators are used as outputs. Thermometers, lights, and other indicators display output values from the program. These may include data, program states, and other information.

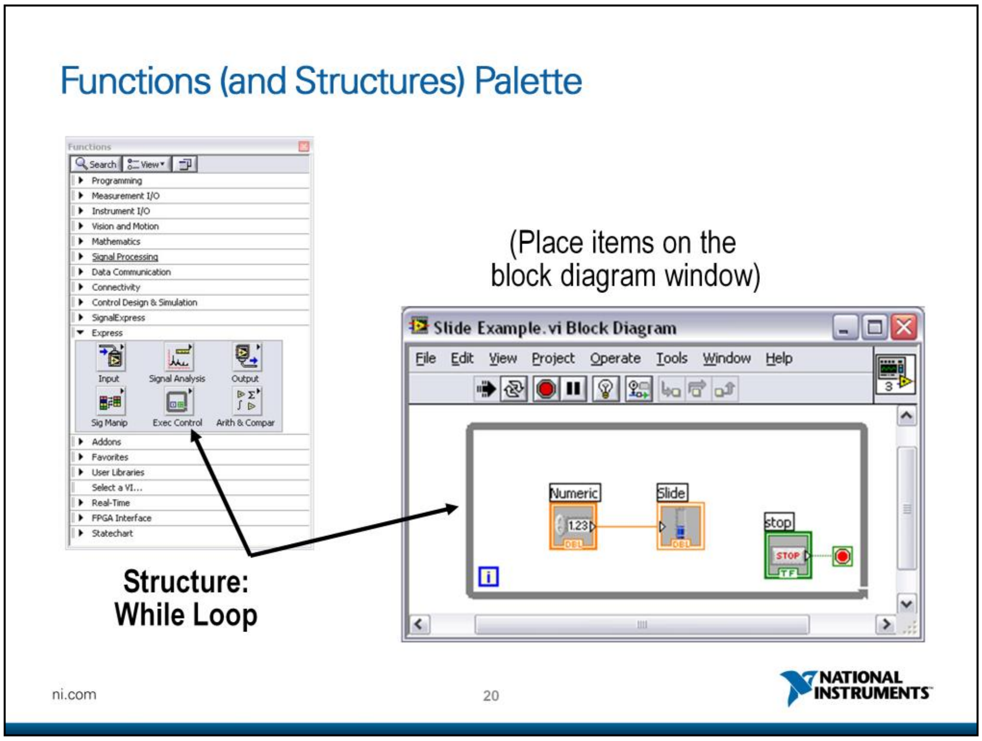
Every front panel control or indicator has a corresponding terminal on the block diagram. When you run a VI, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed into other functions or indicators through wires.



Controls Palette

(Place items on the front panel window)

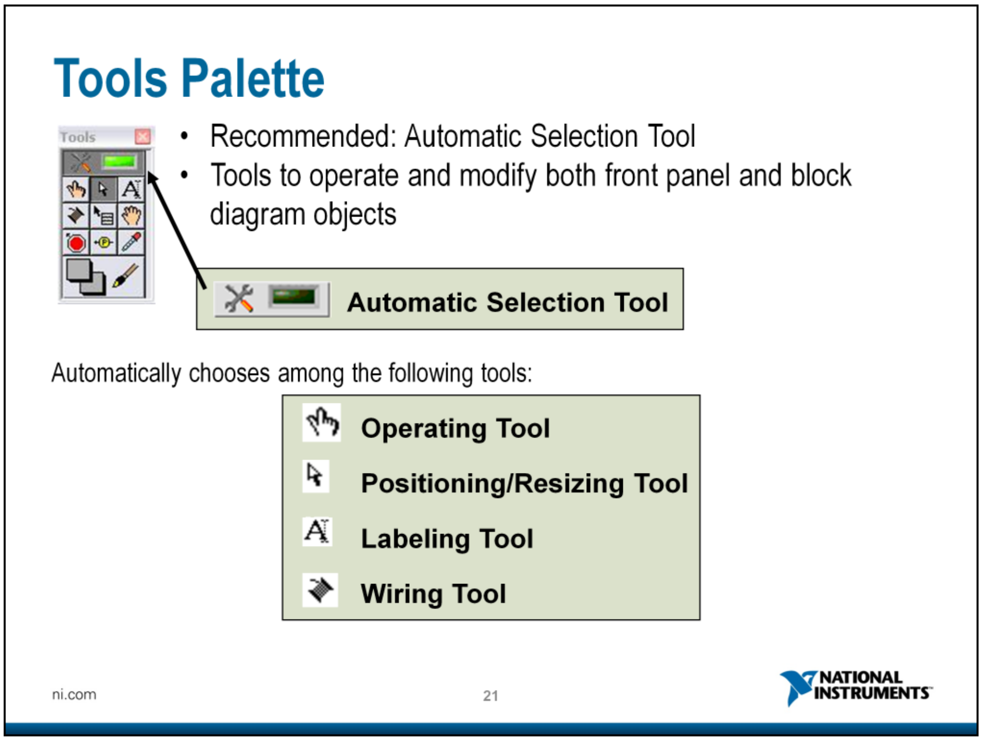
Use the **Controls** palette to place controls and indicators on the front panel. The **Controls** palette is available only on the front panel. To view the palette, select **Window»Show Controls Palette**. You also can display the **Controls** palette by right-clicking an open area on the front panel. Tack down the **Controls** palette by clicking the pushpin on the top left corner of the palette.



Functions (and Structures) Palette

ni.com 20 NATIONAL INSTRUMENTS

Use the **Functions** palette to build the block diagram. The **Functions** palette is available only on the block diagram. To view the palette, select **Window»Show Functions Palette**. You also can display the **Functions** palette by right-clicking an open area on the block diagram. Tack down the **Functions** palette by clicking the pushpin on the top left corner of the palette.



Tools Palette

- Recommended: Automatic Selection Tool
- Tools to operate and modify both front panel and block diagram objects

Automatically chooses among the following tools:

If you enable the automatic selection tool and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the **Tools** palette. Toggle automatic selection tool by clicking the **Automatic Selection Tool** button in the **Tools** palette.

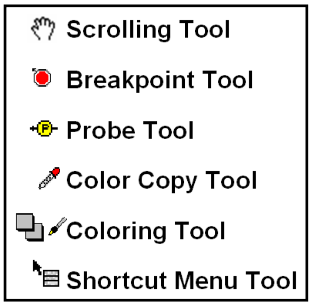
Use the **Operating Tool** to change the values of a control or select the text within a control.

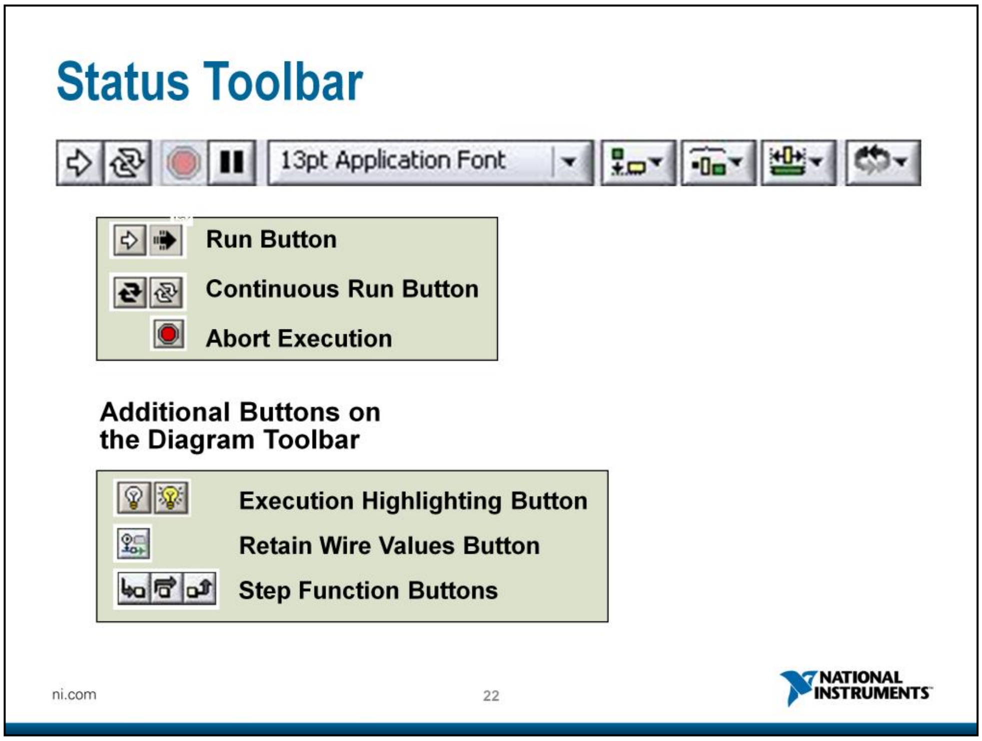
Use the **Positioning/Resizing Tool** to select, move, or resize objects. The Positioning Tool changes shape when it moves over a corner of a resizable object.

Use the **Labeling Tool** to edit text and create free labels. The Labeling Tool changes to a cursor when you create free labels.

Use the **Wiring Tool** to wire objects together on the block diagram.

Other important tools:





Status Toolbar

Run Button

Continuous Run Button

Abort Execution

Additional Buttons on the Diagram Toolbar

Execution Highlighting Button

Retain Wire Values Button

Step Function Buttons

ni.com 22 NATIONAL INSTRUMENTS

- Click the **Run** button to run the VI. While the VI runs, the **Run** button appears with a black arrow if the VI is a top-level VI, meaning it has no callers and therefore is not a subVI.
- Click the **Continuous Run** button to run the VI until you abort or pause it. You also can click the button again to disable continuous running.
- While the VI runs, the **Abort Execution** button appears. Click this button to stop the VI immediately.

Note: Avoid using the **Abort Execution** button to stop a VI. Either let the VI complete its data flow or design a method to stop the VI programmatically. By doing so, the VI is at a known state. For example, place a button on the front panel that stops the VI when you click it.

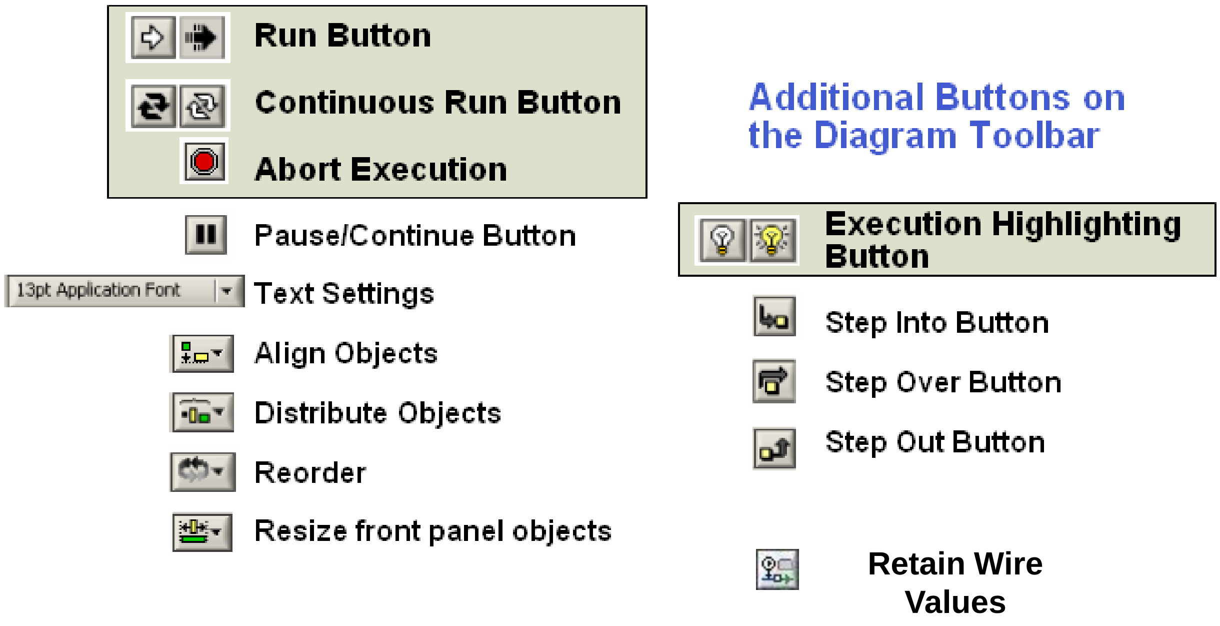
- Click the **Pause** button to pause a running VI. When you click the **Pause** button, LabVIEW highlights on the block diagram the location where you paused execution. Click the **Pause** button again to continue running the VI.
- Select the **Text Settings** pull-down menu to change the font settings for the VI, including size, style, and color.
- Select the **Align Objects** pull-down menu to align objects along axes, including vertical, top edge, left, and so on.
- Select the **Distribute Objects** pull-down menu to space objects evenly, including gaps, compression, and so on.
- Select the **Resize Objects** pull-down menu to change the width and height of front panel objects.

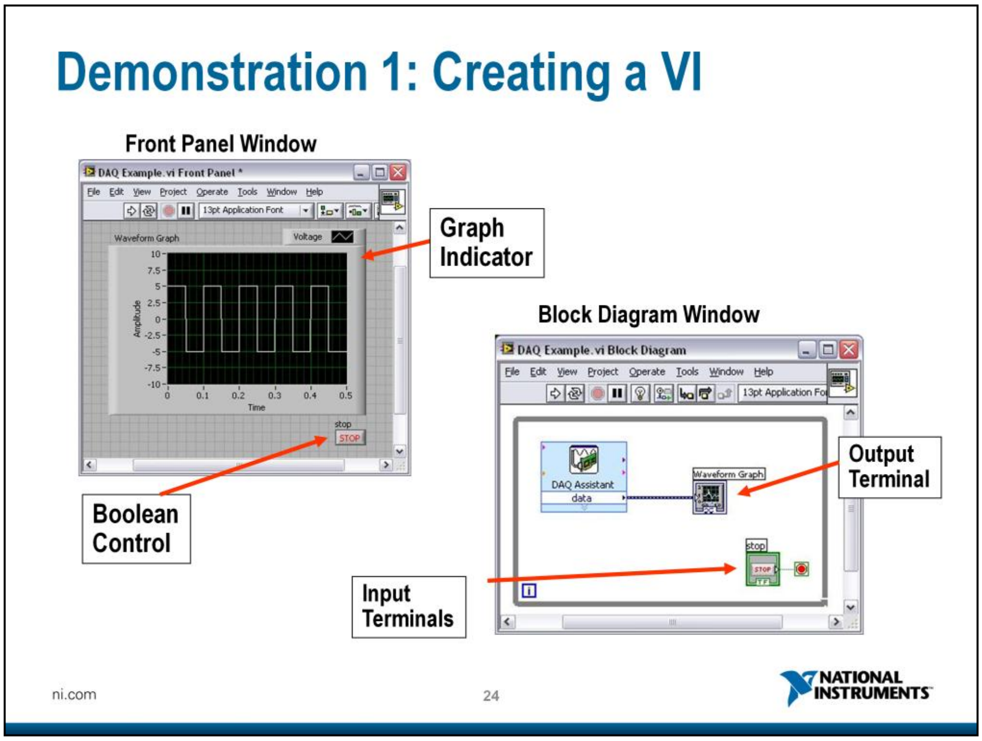
- Select the **Reorder** pull-down menu when you have objects that overlap each other and you want to define which one is in front or back of another. Select one of the objects with the Positioning Tool and then select from **Move Forward**, **Move Backward**, **Move To Front**, and **Move To Back**.

Note: The following items only appear on the block diagram toolbar.

- Click the **Execution Highlighting** button to see the flow of data through the block diagram. Click the button again to disable execution highlighting.
- Click the **Retain Wire Values** button to save the wire values at each point in the flow of execution so that when you place a probe on a wire, you can immediately obtain the most recent value of the data that passed through the wire.
- Click the **Step Into** button to single-step into a loop, subVI, and so on. Single-stepping through a VI steps through the VI node to node. Each node blinks to denote when it is ready to execute. By stepping into the node, you are ready to single-step inside the node.
- Click the **Step Over** button to step over a loop, subVI, and so on. By stepping over the node, you execute the node without single-stepping through the node.
- Click the **Step Out** button to step out of a loop, subVI, and so on. By stepping out of a node, you complete single-stepping through the node and go to the next node.

Additional Tools





Demonstration 1: Creating a VI

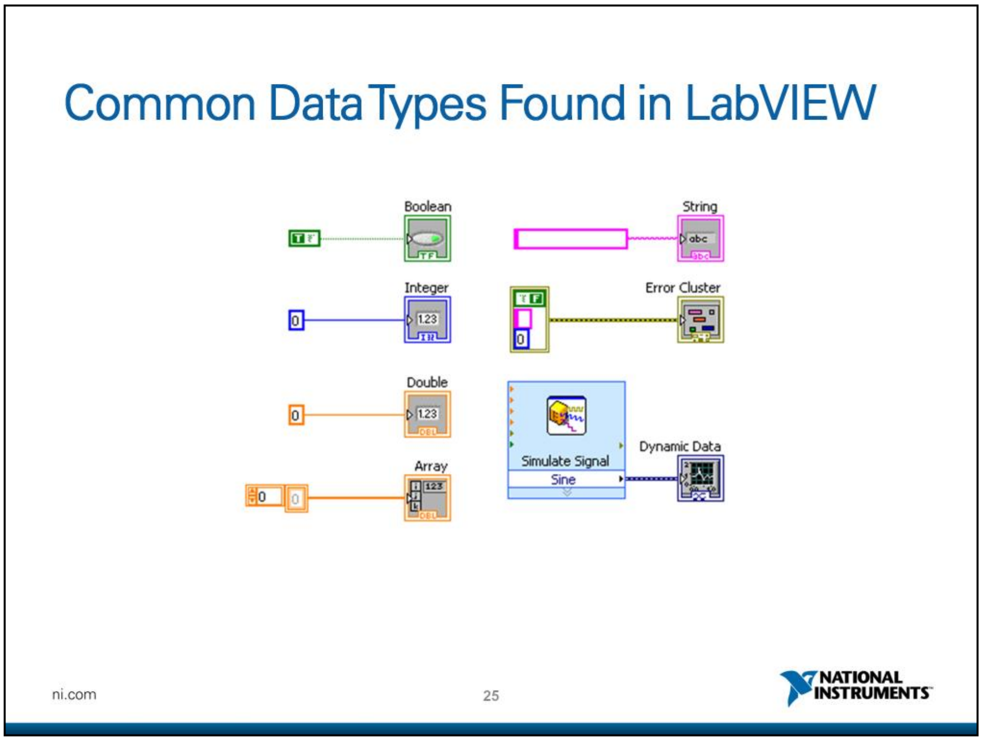
ni.com 24 NATIONAL INSTRUMENTS

When you create an object on the front panel, a terminal is created on the block diagram. These terminals give you access to the front panel objects from the block diagram code.

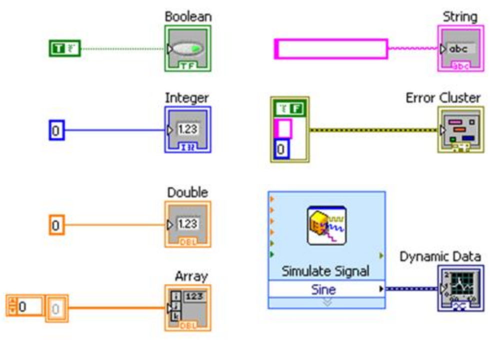
Each terminal contains useful information about the front panel object it corresponds to, and uses color and symbols to provide information about the data type. For example, the dynamic data type is a polymorphic data type represented by dark blue terminals. Boolean terminals are green with TF lettering.

In general, blue terminals should wire to blue terminals, green to green, and so on. This is not a hard-and-fast rule; you can use LabVIEW to connect a blue terminal (dynamic data) to an orange terminal (fractional value), for example. But in most cases, look for a match in colors.

Controls have a thick border and an arrow on the right side. Indicators have a thin border and an arrow on the left side. Logic rules apply to wiring in LabVIEW: Each wire must have one (but only one) source (or control), and each wire may have multiple destinations (or indicators).



Common Data Types Found in LabVIEW



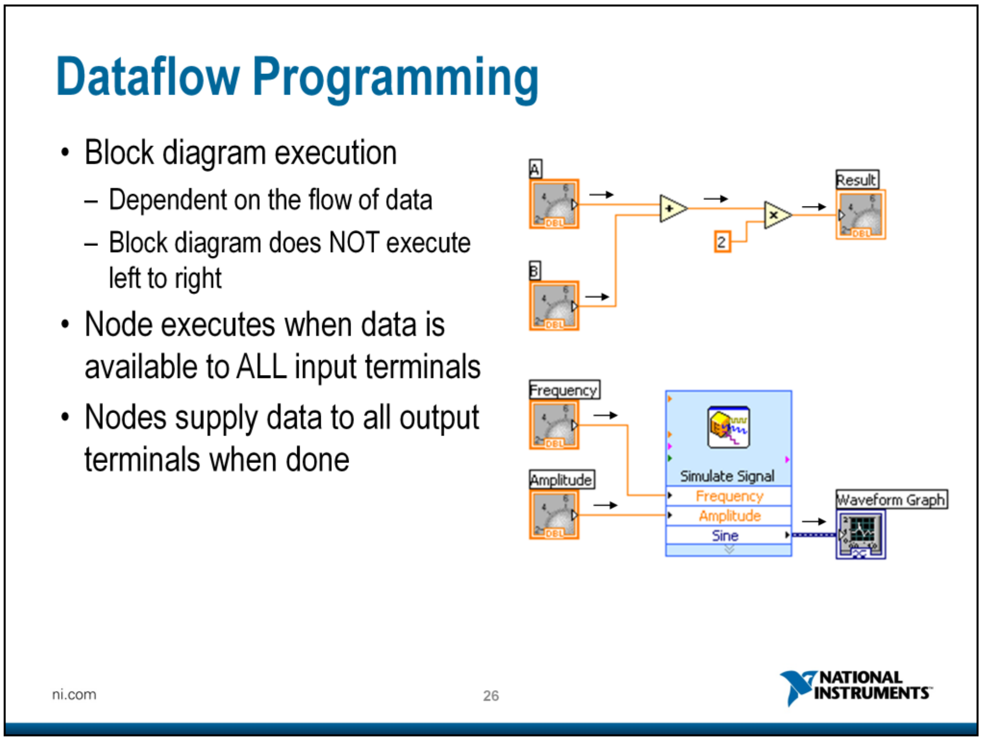
LabVIEW uses many common data types – Boolean, numeric, strings, clusters, and so on.

The color and symbol of each terminal indicate the data type of the control or indicator. Control terminals have a thicker border than indicator terminals. Also, arrows appear on front panel terminals to indicate whether the terminal is a control or an indicator. An arrow appears on the right if the terminal is a control and on the left if the terminal is an indicator.

Definitions

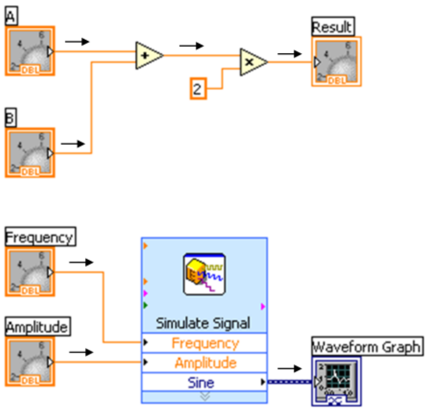
- **Array:** Arrays group data elements of the same type. An array consists of elements and dimensions. Elements are the data that make up the array. A dimension is the length, height, or depth of an array. An array can have one or more dimensions and as many as $(2^{31}) – 1$ elements per dimension, memory permitting.
- **Cluster:** Clusters group data elements of mixed types, much like a bundle of wires in a telephone cable, where each wire in the cable represents a different element of the cluster.

See **Help»Search the LabVIEW Help…** for more information. The *LabVIEW User Manual* on `ni.com` provides additional references for data types found in LabVIEW.



Dataflow Programming

- Block diagram execution
  - Dependent on the flow of data
  - Block diagram does NOT execute left to right
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done

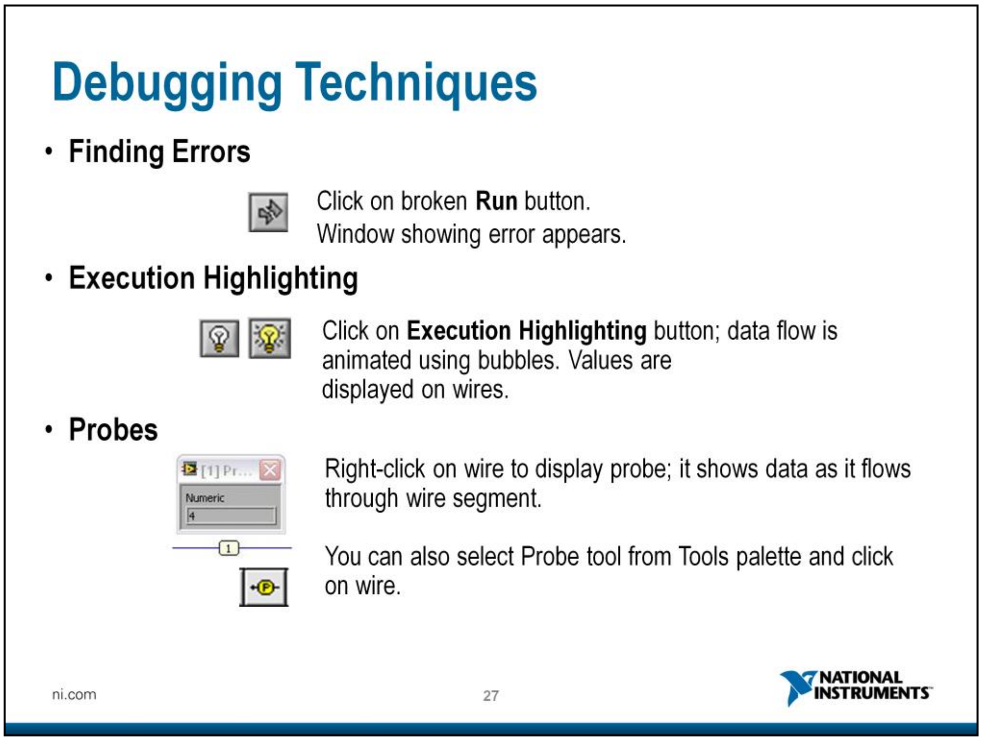


ni.com 26

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when all its inputs are available. When a node completes execution, it supplies data to its output terminals and passes the output data to the next node in the dataflow path. Visual Basic, C++, JAVA, and most other text-based programming languages follow a control flow model of program execution. In control flow, the sequential order of program elements determines the execution order of a program.

Consider the block diagram above. It adds two numbers and then multiplies by 2 from the result of the addition. In this case, the block diagram executes from left to right, not because the objects are placed in that order but because one of the inputs of the Multiply function is not valid until the Add function has finished executing and passed the data to the Multiply function. Remember that a node executes only when data are available at all of its input terminals, and it supplies data to its output terminals only when it finishes execution. In the second piece of code, the Simulate Signal Express VI receives input from the controls and passes its result to the graph.

You may consider the add-multiply and the simulate signal code to coexist on the same block diagram in parallel. This means that they begin executing at the same time and run independently of one another. If the computer running this code had multiple processors, these two pieces of code could run independently of one another (each on its own processor) without any additional coding.



Debugging Techniques

- **Finding Errors**

  Click on broken **Run** button.
  Window showing error appears.

- **Execution Highlighting**

  Click on **Execution Highlighting** button; data flow is animated using bubbles. Values are displayed on wires.

- **Probes**

  Right-click on wire to display probe; it shows data as it flows through wire segment.

  You can also select Probe tool from Tools palette and click on wire.

ni.com 27 NATIONAL INSTRUMENTS

When your VI is not executable, a broken arrow is displayed in the **Run** button in the palette.

- **Finding Errors**: To list errors, click on the broken arrow. To locate the bad object, click on the error message.
- **Execution Highlighting**: Animates the diagram and traces the flow of the data, allowing you to view intermediate values. Click on the **light bulb** on the toolbar.
- **Probe**: Used to view values in arrays and clusters. Click on wires with the **Probe** tool or right-click on the wire to set probes.
- **Retain Wire Values**: Used with probes to view the values from the last iteration of the program.
- **Breakpoint**: Sets pauses at different locations on the diagram. Click on wires or objects with the **Breakpoint** tool to set breakpoints.

Exercise 2 – Acquiring a Signal with DAQ (Track A)

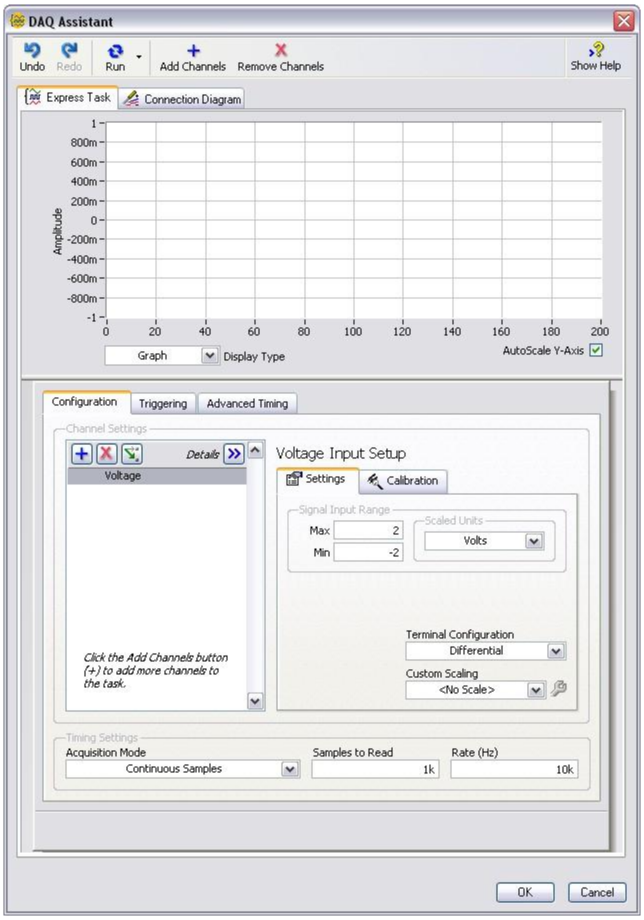
Complete the following steps to create a VI that acquires data continuously from your DAQ device.

1. Launch LabVIEW.
2. In the **Getting Started** window, click the **File»New** or **More …** link to display the **New** dialog box.
3. Open a data acquisition template. From the Create New list, select **VI**»**From Templat**e**»DAQ»Data Acquisition with NI-DAQmx.vi** and click **OK**.
4. Display the block diagram by clicking it or by selecting **Window»Show Block Diagram**. Read the instructions written there about how to complete the program.
5. Double-click the DAQ Assistant to launch the configuration wizard.
6. Configure an analog input operation.
   a. Choose **Acquire Signals»Analog Input»Voltage**.
   b. Choose **Dev1 (USB-6009)»ai0** to acquire data on analog input channel 0 and click **Finish**.
   c. In the next window, define the parameters of your analog input operation. To choose an input range that works well with your microphone, on the Settings tab enter `2 V` for the maximum and `–2 V` for the minimum. Under timing settings choose "Continuous" for the acquisition mode and enter `10000` for the rate. Leave all other choices set to their default values. Click **OK** to exit the wizard.
7. Place the Filter Express VI to the right of the DAQ Assistant on the block diagram. From the Functions palette, select **Express»Signal Analysis**»**Filter** and place it on the block diagram inside the While Loop. When you bring up the Functions palette, press the small pushpin in the upper left-hand corner of the palette. This tacks down the palette so that it remains visible. This step is omitted in the following exercises, but you should repeat it. In the configuration window under **Filtering Type**, choose "Highpass." Under **Cutoff Frequency**, use a value of "300 Hz." Click **OK**.

8. Make the following connections on the block diagram by hovering your mouse over the terminal so that it becomes the wiring tool and clicking once on each of the terminals you wish to connect:
    a. Connect the **Data** output terminal of the DAQ Assistant VI to the **Signal** input of the Filter VI.
    b. Create a graph indicator for the filtered signal by right-clicking on the **Filtered Signal** output terminal and choosing **Create»Graph Indicator**.
9. Return to the front panel by selecting **Window»Show Front Panel** or by pressing <Ctrl-E>.
10. Run your program by clicking the **Run** button. Hum or whistle into the microphone to observe the changing voltage on the graph.
11. Click **Stop** once you are finished.
12. Save the VI as `Exercise 2 - Acquire.vi` in your Exercises folder and close it.

Note: The solution to this exercise is printed in the back of this manual.

Tip: You can place the DAQ Assistant on your block diagram from the Functions palette. Right-click the block diagram to open the Functions palette and go to **Express»Input** to find it.



Signal Input Range
Max 2
Min -2

Scaled Units: Volts

Terminal Configuration: Differential

Custom Scaling: <No Scale>

Click the Add Channels button (+) to add more channels to the task.

Timing Settings
Acquisition Mode: Continuous Samples
Samples to Read: 1k
Rate (Hz): 10k

End of Exercise 2 (Track A)

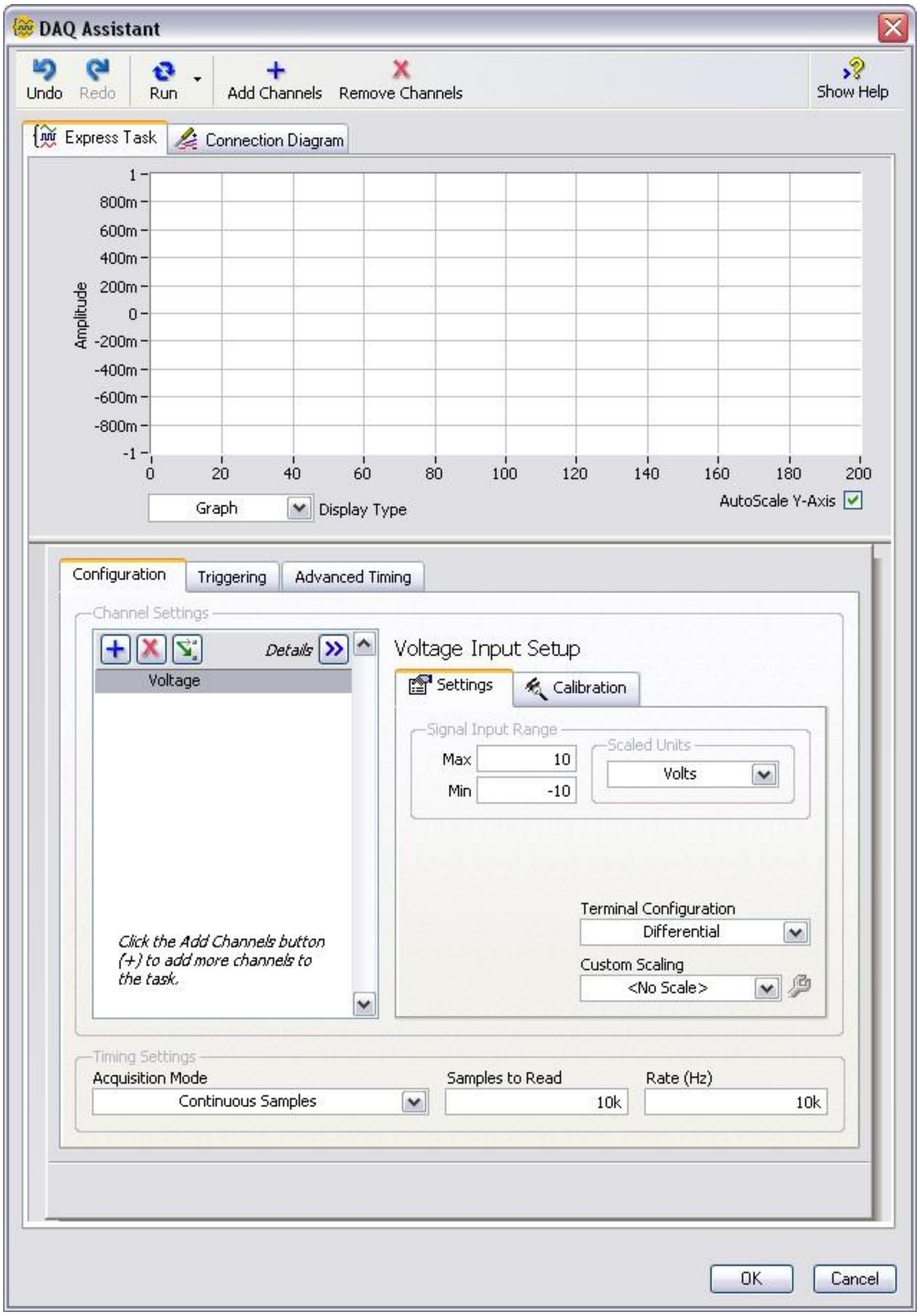
Exercise 2 – Acquiring a Signal with DAQ (Track B)

Complete the following steps to create a VI that acquires data continuously from your simulated DAQ device.

1. Launch LabVIEW.
2. In the **Getting Started** window, click the **File»New** or **More …** link to display the **New** dialog box.
3. Open a data acquisition template. From the Create New list, select **VI»From Template»DAQ»Data Acquisition with NI-DAQmx.vi** and click **OK**.
4. Display the block diagram by clicking it or by selecting **Window»Show Block Diagram**. Read the instructions written there about how to complete the program.
5. Double-click the DAQ Assistant to launch the configuration wizard.
6. Configure an analog input operation.
   a. Choose **Acquire Signals»Analog Input»Voltage**.
   b. Choose **Dev1 (PCI-6220)»ai0** to acquire data on analog input channel 0 and click **Finish**.
   c. In the next window, define the parameters of your analog input operation. On the task timing tab, choose "Continuous" for the acquisition mode, enter `10000` for samples to read, and `10000` for the rate. Leave all the other choices set to their default values. Click **OK** to exit the wizard.
7. Create a graph indicator for the signal by right-clicking on the "Data" output on the DAQ Assistant and choosing **Create»Graph Indicator**.
8. Return to the front panel by selecting **Window»Show Front Panel** or by pressing <Ctrl-E>.
9. Run your program by clicking the run button. Observe the simulated sine wave on the graph.
10. Click **Stop** once you are finished.
11. Save the VI as `Exercise 2 – Acquire.vi` in the Exercises folder. Close the VI.

Notes:

- The solution to this exercise is printed in the back of this manual.
- You can place the DAQ Assistant on your block diagram from the **Functions** palette. Right-click the block diagram to open the **Functions** palette and go to **Express»Input** to find it. When you bring up the **Functions** palette, press the small pushpin in the upper left-hand corner of the palette. This tacks down the palette so that it doesn't disappear. This step is omitted in the following exercises, but you should repeat it.



Voltage

Click the Add Channels button (+) to add more channels to the task.

Voltage Input Setup

Max 10
Min -10
Volts

Terminal Configuration: Differential
Custom Scaling: <No Scale>

Timing Settings
Acquisition Mode: Continuous Samples
Samples to Read: 10k
Rate (Hz): 10k

End of Exercise 2 (Track B)

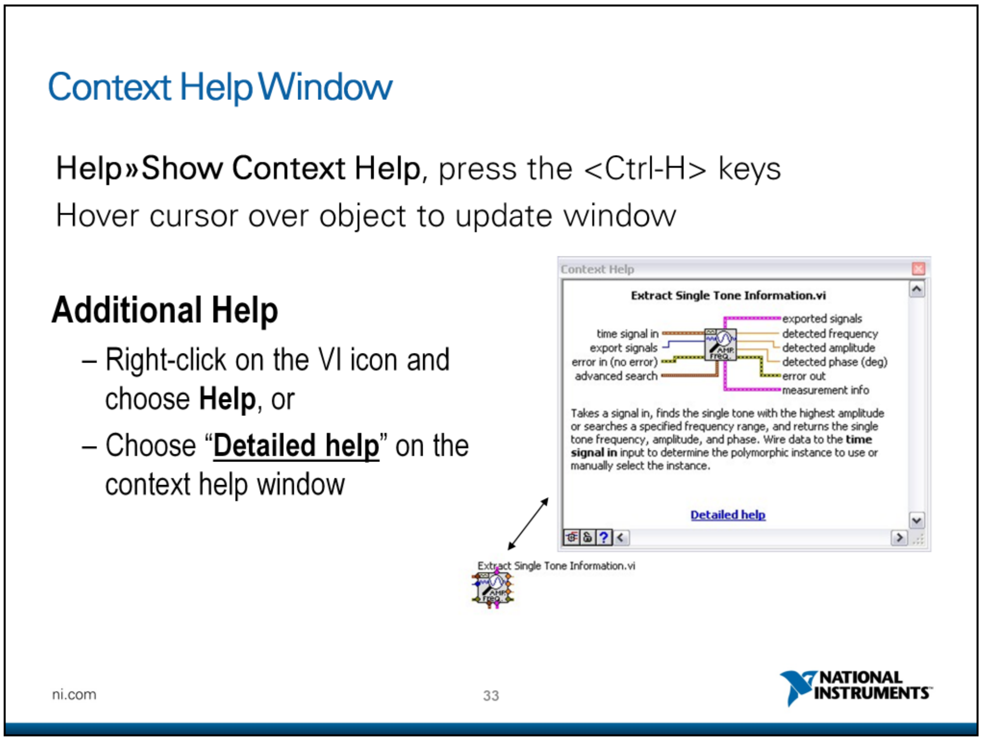
Exercise 2 – Acquiring a Signal with the Sound Card (Track C)

Complete the following steps to create a VI that acquires data from your sound card.

1. Launch LabVIEW.
2. In the **Getting Started** window, click the **Blank VI** link.
3. Display the block diagram by pressing <Ctrl-E> or selecting **Window»Show Block Diagram**.
4. Place the Acquire Sound Express VI on the block diagram. Right-click to open the **functions** palette and select **Express»Input»Acquire Sound.** Place the Express VI on the block diagram.
5. In the configuration window under **#Channels**, select **1** from the pull-down list. Under **Duration(s)**, use a value of **5 seconds**. Click **OK**.
6. Place the Filter Express VI to the right of the Acquire Signal VI on the block diagram. From the **Functions** palette, select **Express»Signal Analysis**»**Filter** and place it on the block diagram. In the configuration window under **Filtering Type**, choose "Highpass." Under **Cutoff Frequency**, use a value of 300 Hz. Click **OK**.
7. Make the following connections on the block diagram by hovering your mouse over the terminal so that it becomes the wiring tool and clicking once on each of the terminals you wish to connect:
    a. Connect the **Data** output terminal of the Acquire Sound VI to the **Signal** input of the Filter VI.
    b. Create a graph indicator for the filtered signal by right-clicking on the **Filtered Signal** output terminal and choose **Create**»**Graph Indicator**.
8. Return to the front panel by pressing <Ctrl-E> or **Window»Show Front Panel**.
9. Run your program by clicking the **Run** button. Hum or whistle into your microphone and observe the data you acquire from your sound card.
10. Save the VI as `Exercise 2 - Acquire.vi` in the Exercises folder.
11. Close the VI.

Note: The solution to this exercise is printed in the back of this manual.

End of Exercise (Track C)



Context Help Window

Help»Show Context Help, press the <Ctrl-H> keys

Hover cursor over object to update window

Additional Help

- Right-click on the VI icon and choose **Help**, or
- Choose "**Detailed help**" on the context help window

Context Help

Extract Single Tone Information.vi

Takes a signal in, finds the single tone with the highest amplitude or searches a specified frequency range, and returns the single tone frequency, amplitude, and phase. Wire data to the **time signal in** input to determine the polymorphic instance to use or manually select the instance.

Detailed help

Extract Single Tone Information.vi

ni.com 33 NATIONAL INSTRUMENTS

The **Context Help** window displays basic information about LabVIEW objects when you move the cursor over each object. Objects with context help information include VIs, functions, constants, structures, palettes, properties, methods, events, and dialog box components.

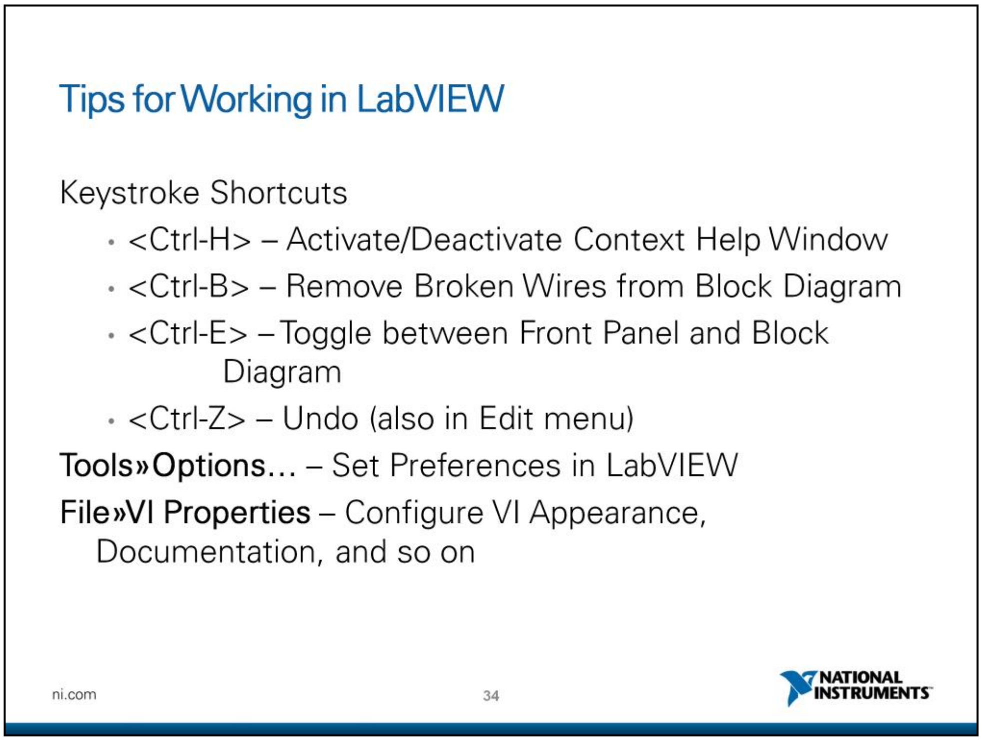
To display the **Context Help** window, select **Help»Show Context Help**, press the <Ctrl-H> keys, or press the **Show Context Help Window** button in the toolbar.

Connections displayed in Context Help:

Required – bold
Recommended – normal
Optional – dimmed

Additional Help

- **VI, Function, & How-To Help is also available**
  - **Help»VI, Function, & How-To Help**
  - Right-click the VI icon and choose **Help**, or
  - Choose "Detailed Help" on the **Context Help** window
- **LabVIEW Help – reference style help**
  - **Help»Search the LabVIEW Help…**



Tips for Working in LabVIEW

Keystroke Shortcuts

- <Ctrl-H> – Activate/Deactivate Context Help Window
- <Ctrl-B> – Remove Broken Wires from Block Diagram
- <Ctrl-E> – Toggle between Front Panel and Block Diagram
- <Ctrl-Z> – Undo (also in Edit menu)

Tools»Options… – Set Preferences in LabVIEW

File»VI Properties – Configure VI Appearance, Documentation, and so on

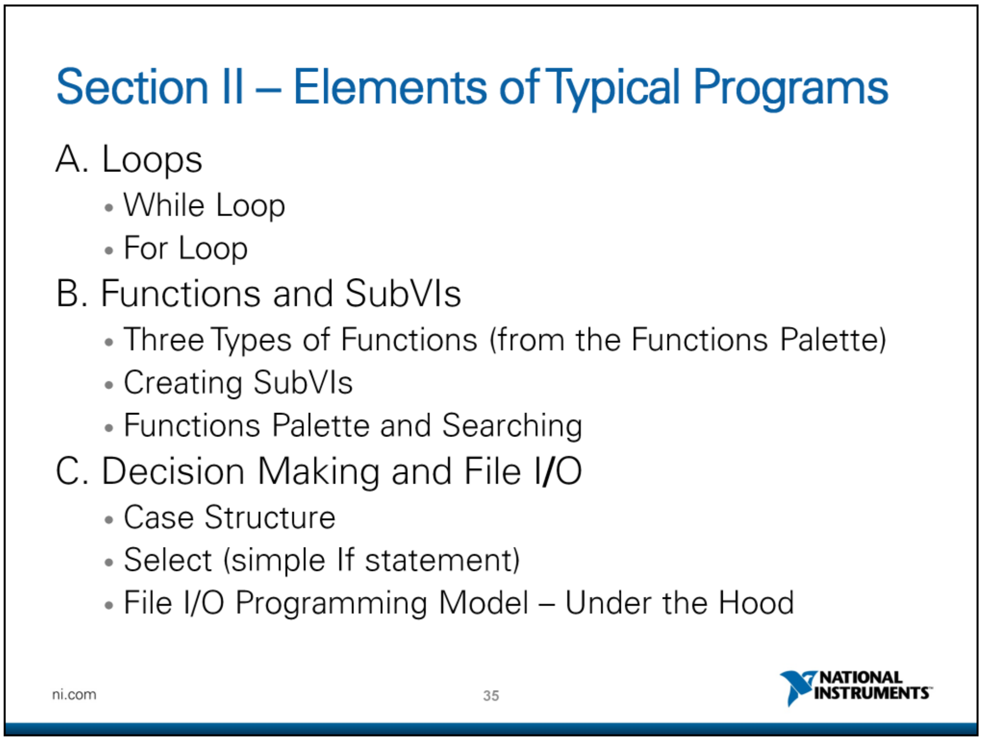
ni.com 34

LabVIEW has many keystroke shortcuts that make working easier. The most common shortcuts are listed above.

While the Automatic Selection Tool is great for choosing the tool you would like to use in LabVIEW, there are sometimes cases when you want manual control. Once the Automatic Selection Tool is turned off, use the Tab key to toggle between the four most common tools (Operate Value, Position/Size/Select, Edit Text, Set Color on front panel and Operate Value, Position/Size/Select, Edit Text, Connect Wire on block diagram). Once you are finished with the tool you chose, you can press <Shift-Tab> to turn the Automatic Selection Tool back on.

In the **Tools»Options…** dialog, there are many configurable options for customizing your front panel, block diagram, colors, printing, and so on.

Similar to the LabVIEW options, you can configure VI-specific properties by going to **File»VI Properties…** .There you can document the VI, change the appearance of the window, and customize it in several other ways.

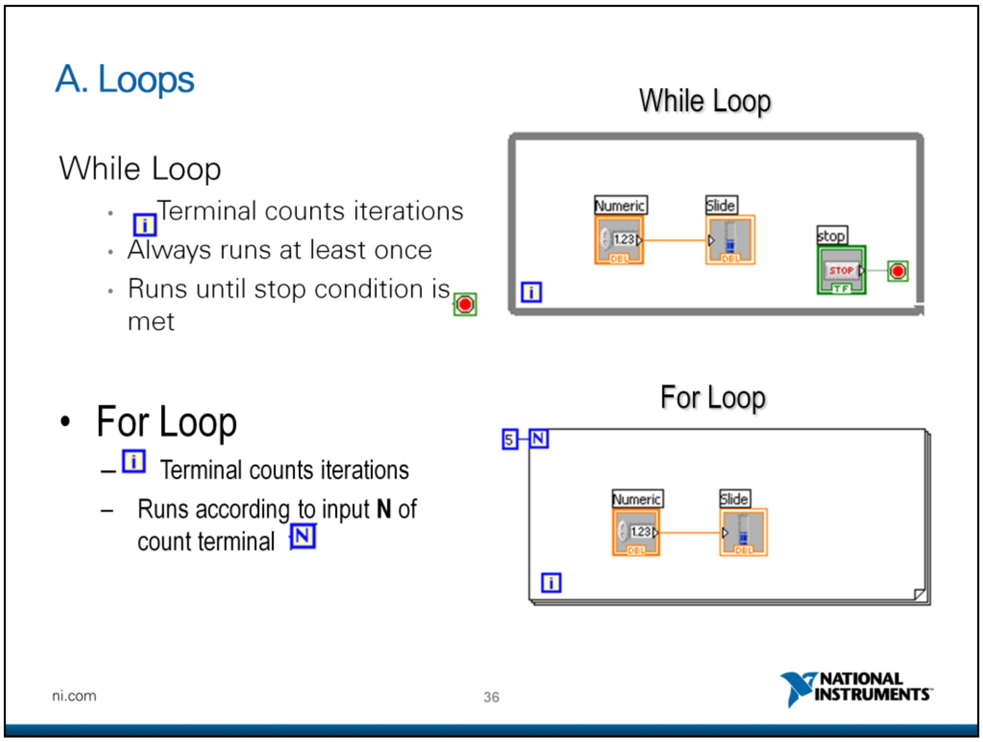


Section II – Elements of Typical Programs

A. Loops
- While Loop
- For Loop

B. Functions and SubVIs
- Three Types of Functions (from the Functions Palette)
- Creating SubVIs
- Functions Palette and Searching

C. Decision Making and File I/O
- Case Structure
- Select (simple If statement)
- File I/O Programming Model – Under the Hood



A. Loops

While Loop

- $\boxed{i}$ Terminal counts iterations
- Always runs at least once
- Runs until stop condition is met

- For Loop
  - $\boxed{i}$ Terminal counts iterations
  - Runs according to input **N** of count terminal $\boxed{N}$

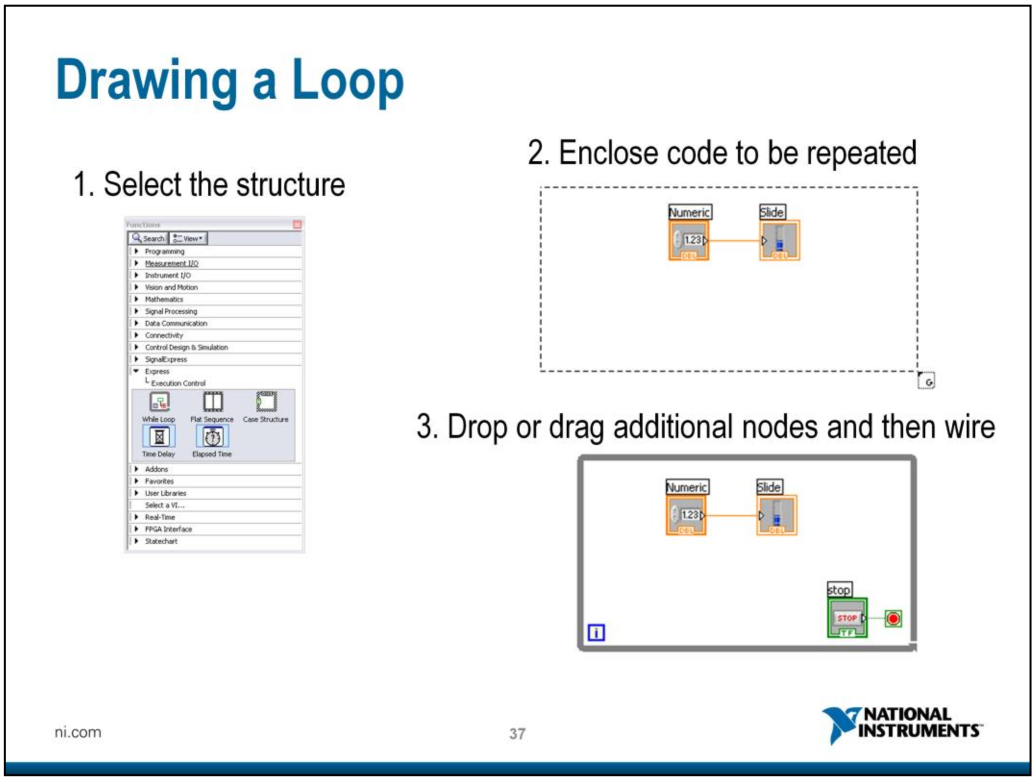
Both the While and For Loops are located on the **Functions»Structures** palette. The For Loop differs from the While Loop in that the For Loop executes a set number of times. A While Loop stops executing the subdiagram only if the value at the conditional terminal exists.

While Loops

Similar to a Do loop or a Repeat-until loop in text-based programming languages, a While Loop, shown at the top right, executes a subdiagram until a condition is met. The While Loop executes the subdiagram until the conditional terminal, an input terminal, receives a specific Boolean value. The default behavior and appearance of the conditional terminal is **Stop If True**. When a conditional terminal is **Stop If True**, the While Loop executes its subdiagram until the conditional terminal receives a TRUE value. The iteration terminal $\boxed{i}$ (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.

For Loops

A For Loop, shown above, executes a subdiagram a set number of times. The value in the count terminal (an input terminal) represented by the N, indicates how many times to repeat the subdiagram. The iteration terminal $\boxed{i}$ (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.



Drawing a Loop

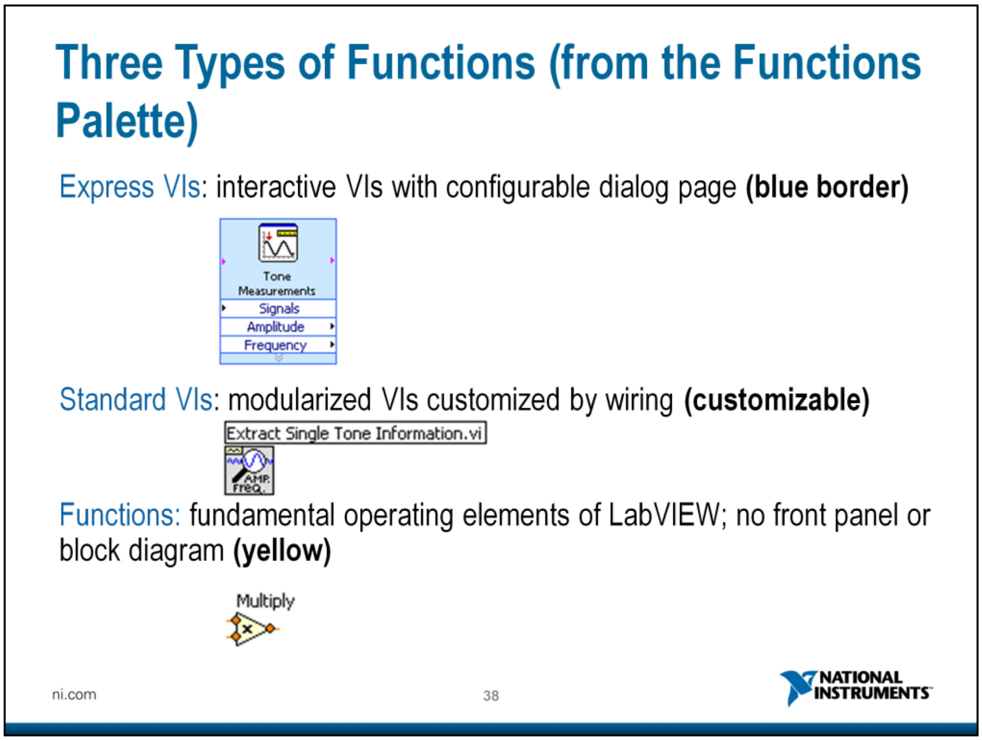
1. Select the structure

2. Enclose code to be repeated

3. Drop or drag additional nodes and then wire

Place loops in your diagram by selecting them from the **Functions** palette:

- When selected, the mouse cursor becomes a special pointer that you use to enclose the section of code you want to include in the While Loop.
- Click the mouse button to define the top-left corner and then click the mouse button again at the bottom-right corner. The While Loop boundary appears around the selected code.
- Drag or drop additional nodes in the While Loop if needed.



Three Types of Functions (from the Functions Palette)

Express VIs: interactive VIs with configurable dialog page **(blue border)**

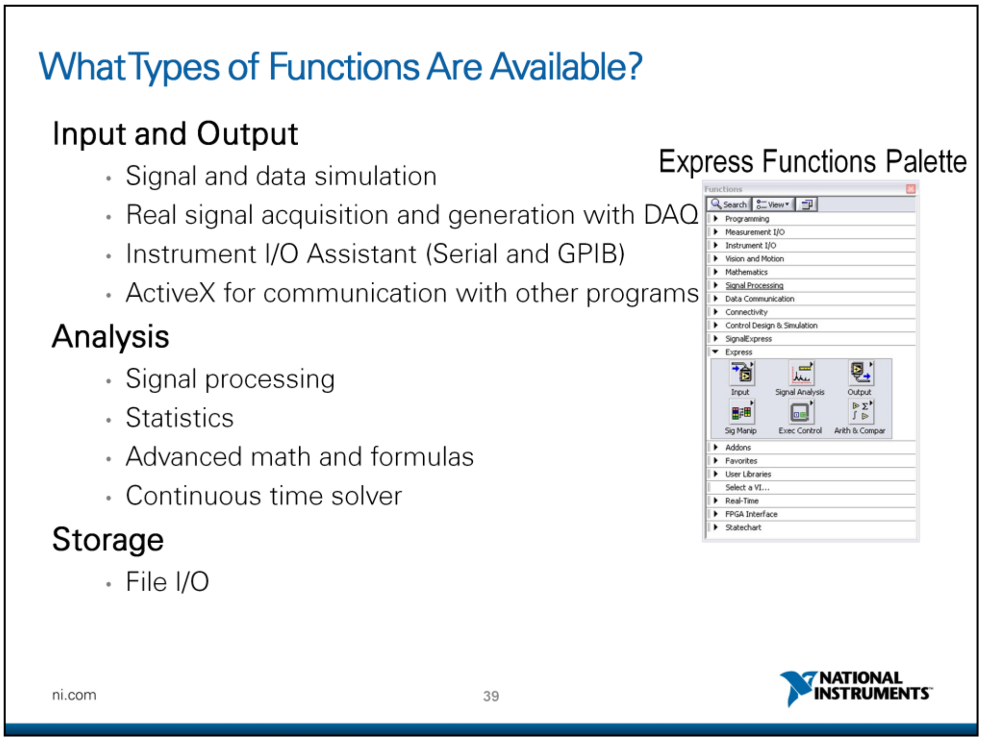
Standard VIs: modularized VIs customized by wiring **(customizable)**

Functions: fundamental operating elements of LabVIEW; no front panel or block diagram **(yellow)**

LabVIEW 7.0 introduced a new type of subVI called Express VIs. These are interactive VIs that have a configuration dialog box that helps the user customize the functionality of the Express VI. LabVIEW then generates a subVI based on these settings.

SubVIs are VIs (consisting of a front panel and a block diagram) that are used within another VI.

Functions are the building blocks of all VIs. Functions do not have a front panel or a block diagram.



What Types of Functions Are Available?

Input and Output
- Signal and data simulation
- Real signal acquisition and generation with DAQ
- Instrument I/O Assistant (Serial and GPIB)
- ActiveX for communication with other programs

Analysis
- Signal processing
- Statistics
- Advanced math and formulas
- Continuous time solver

Storage
- File I/O

Express Functions Palette

Functions
Search
View
Programming
Measurement I/O
Instrument I/O
Vision and Motion
Mathematics
Signal Processing
Data Communication
Connectivity
Control Design & Simulation
SignalExpress
Express
Input
Signal Analysis
Output
Sig Manip
Exec Control
Arith & Compar
Addons
Favorites
User Libraries
Select a VI...
Real-Time
FPGA Interface
Statechart

ni.com 39 NATIONAL INSTRUMENTS

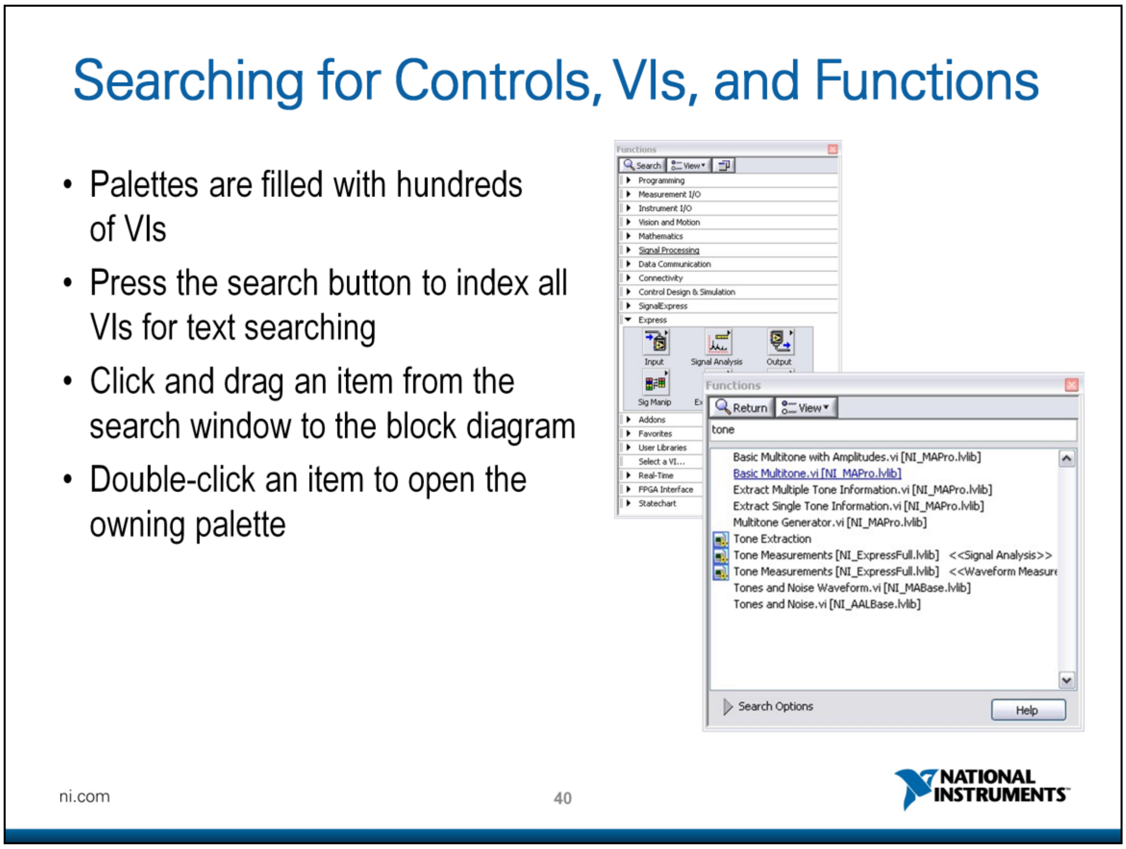
LabVIEW includes several hundred prebuilt functions to help you acquire, analyze, and present data. You would generally use these functions as outlined on the slide above.

LabVIEW Toolkits

Additional toolkits are available for adding domain-specific functionality to LabVIEW. These toolkits include:

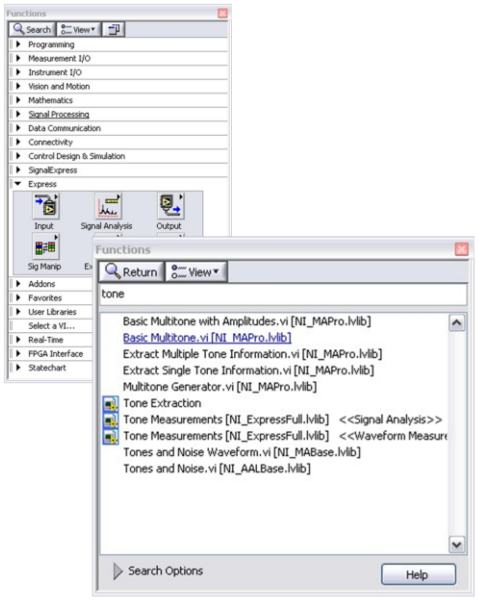
Application Deployment and Targeting Modules	**Signal Processing and Analysis**	**Control Design and Simulation**
* LabVIEW Mobile Module * LabVIEW Real-Time Module * LabVIEW FPGA Module * NI Vision Development Module for LabVIEW	* Sound and Vibration Toolkit * Advanced Signal Processing Toolkit * Modulation Toolkit * Spectral Measurements Toolkit * Order Analysis Toolkit * Digital Filter Design Toolkit	* LabVIEW Control Design and Simulation Module * LabVIEW Real-Time Module * System Identification Toolkit * State Diagram Toolkit
Embedded System Deployment	**Software Engineering and Optimization Tools**	**Image Processing and Acquisition**
* DSP Test Integration Toolkit * Embedded test integration toolkits * Digital Filter Design Toolkit * LabVIEW FPGA Module	* Real-Time Execution Trace Toolkit * Express VI Development Toolkit * State Diagram Toolkit * VI Analyzer Toolkit	* Vision Development Module for LabVIEW * NI Vision Builder for Automated Inspection * NI-IMAQ for IEEE 1394

ni.com/toolkits



Searching for Controls, VIs, and Functions

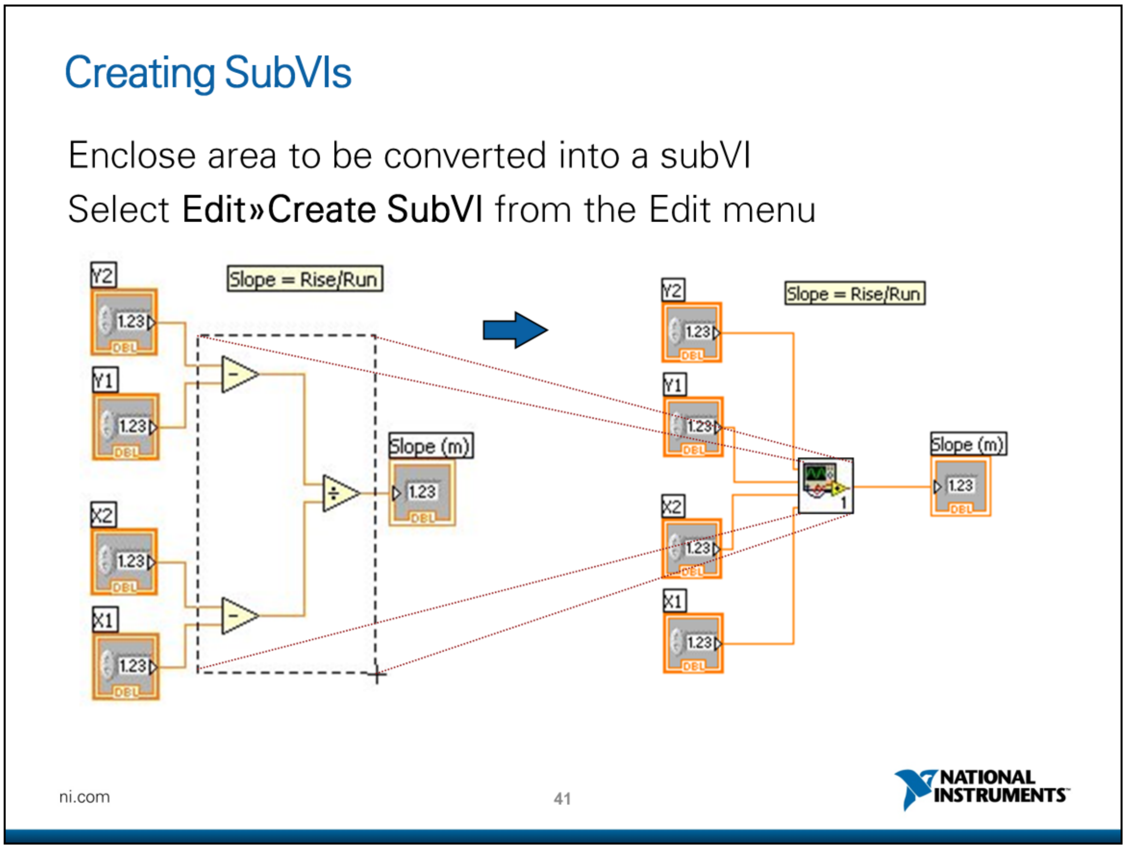
- Palettes are filled with hundreds of VIs
- Press the search button to index all VIs for text searching
- Click and drag an item from the search window to the block diagram
- Double-click an item to open the owning palette



```
Basic Multitone with Amplitudes.vi [NI_MAPro.lvlib]
Basic Multitone.vi [NI_MAPro.lvlib]
Extract Multiple Tone Information.vi [NI_MAPro.lvlib]
Extract Single Tone Information.vi [NI_MAPro.lvlib]
Multitone Generator.vi [NI_MAPro.lvlib]
Tone Extraction
Tone Measurements [NI_ExpressFull.lvlib]   <<Signal Analysis>>
Tone Measurements [NI_ExpressFull.lvlib]   <<Waveform Measure
Tones and Noise Waveform.vi [NI_MABase.lvlib]
Tones and Noise.vi [NI_AALBase.lvlib]
```

Use the buttons on top of the palette windows to navigate, search, and edit the palettes.

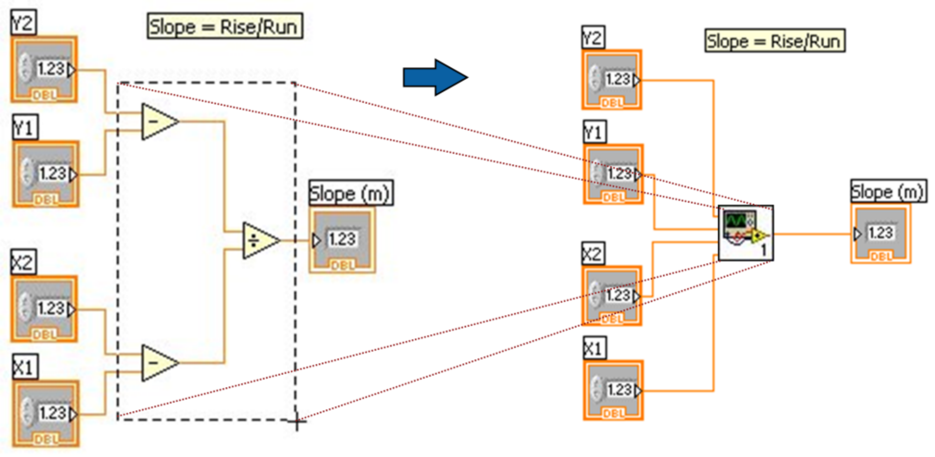
You can search for controls, VIs, and functions that either contain certain words or start with certain words. Double-clicking a search result opens the palette that contains the search result. You also can click and drag the name of the control, VI, or function directly to the front panel or block diagram.



Creating SubVIs

Enclose area to be converted into a subVI

Select **Edit»Create SubVI** from the Edit menu



ni.com 41 NATIONAL INSTRUMENTS

Creating SubVIs

After you build a VI, you can use it in another VI. A VI called from the block diagram of another VI is called a subVI. You can reuse a subVI in other VIs. To create a subVI, you need to build a connector pane and create an icon.

A subVI node corresponds to a subroutine call in text-based programming languages. A block diagram that contains several identical subVI nodes calls the same subVI several times.

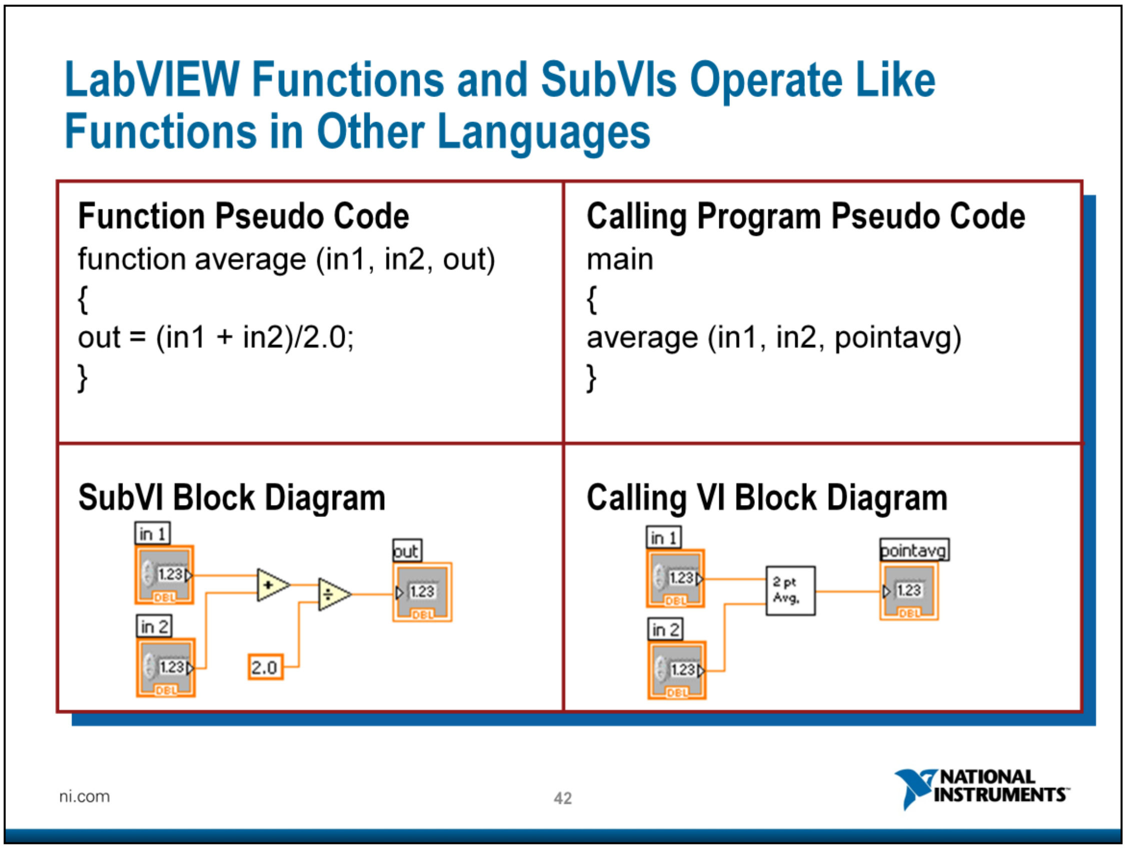
The subVI controls and indicators receive data from and return data to the block diagram of the calling VI. Click the **Select a VI** icon or text on the **Functions** palette, navigate to and double-click a VI, and place the VI on a block diagram to create a subVI call to that VI.

You can easily customize subVI input and output terminals as well as the icon. Follow the instructions below to quickly create a subVI.

Creating SubVIs from Sections of a VI

Convert a section of a VI into a subVI by using the Positioning tool to select the section of the block diagram you want to reuse and selecting **Edit»Create SubVI**. An icon for the new subVI replaces the selected section of the block diagram. LabVIEW creates controls and indicators for the new subVI, automatically configures the connector pane based on the number of control and indicator terminals you selected, and wires the subVI to the existing wires.

See **Help»Search the LabVIEW Help…»SubVIs** for more information.



LabVIEW Functions and SubVIs Operate Like Functions in Other Languages

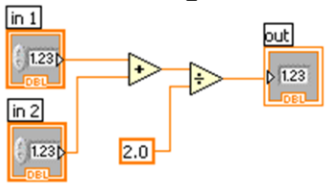
Function Pseudo Code

```
function average (in1, in2, out)
{
out = (in1 + in2)/2.0;
}
```

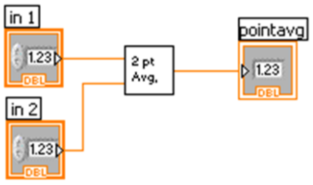
Calling Program Pseudo Code

```
main
{
average (in1, in2, pointavg)
}
```

SubVI Block Diagram



Calling VI Block Diagram



A subVI node corresponds to a subroutine call in text-based programming languages. The node is not the subVI itself, just as a subroutine call statement in a program is not the subroutine itself. A block diagram that contains several identical subVI nodes calls the same subVI several times.

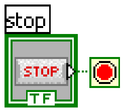
The modular approach makes applications easier to debug and maintain.

The functionality of the subVI does not matter for this example. The important point is the passing of two numeric inputs and one numeric output.

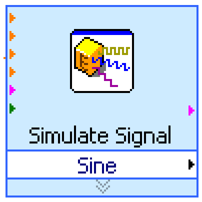
Exercise 3.1 – Analysis (Tracks A, B, and C)

Create a VI that produces a sine wave with a specified frequency and displays the data on a waveform chart until stopped by the user.

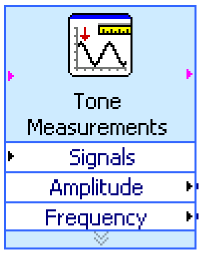
1. Open a blank VI from the Getting Started screen.
2. Place a chart on the front panel. Right-click to open the **Controls** palette and select **Controls»Modern»Graph»Waveform Chart**.
3. Place a dial control on the front panel. From the **Controls** palette, select **Controls»Modern»Numeric»Dial**. Notice that when you first place the control on the front panel, the label text is highlighted. While this text is highlighted, type `Frequency In` to give a name to this control.
4. Go to the block diagram (<Ctrl-E>) and put down a While Loop. Right-click to open the **Functions** palette and select **Express»Execution Control»While Loop**. Click and drag on the block diagram to make the While Loop the correct size. Select the waveform chart and dial and drag them inside the While Loop if they are not already. Notice that a **Stop** button is already connected to the conditional terminal of the While Loop.



1. Place the Simulate Signal Express VI on the block diagram. From the **Functions** palette, select **Express»Signal Analysis»Simulate Signal** and place it on the block diagram inside the While Loop. In the configuration window under Timing, choose "Simulate acquisition timing." Click **OK**.



6. Place a Tone Measurements Express VI on the block diagram (**Express»Signal Analysis»Tone).** In the configuration window, choose **Amplitude** and **Frequency** measurements in the Single Tone Measurements section. Click **OK**.



7. Make the following connections on the block diagram by hovering your mouse over the terminal so that it becomes the wiring tool and clicking once on each of the terminals you wish to connect:
   a. Connect the "Sine" output terminal of the Simulate Signal VI to the "Signals" input of the Tone Measurements VI.
   b. Connect the "Sine" output to the waveform chart.
   c. Create indicators for the amplitude and frequency measurements by right-clicking on each of the terminals of the Tone Measurements Express VI and selecting **Create»Numeric Indicator**.
   d. Connect the "Frequency In" control to the "Frequency" terminal of the Simulate Signal VI.
8. Return to the front panel and run the VI. Move the "Frequency In" dial and observe the frequency of the signal. Click the Stop button once you are finished.
9. Save the VI as `Exercise 3.1 – Simulated.vi`.
10. Close the VI.

Notes

- When you bring up the **Functions** palette, press the small pushpin in the upper left-hand corner of the palette. This tacks down the palette so that it doesn't disappear. This step is omitted in the following exercises, but you should repeat it.
- The solution to this exercise is printed in the back of this manual.

End of Exercise 3.1 (Tracks A, B, and C)

Exercise 3.2 – Analysis (Track A)

Create a VI that measures the frequency of a filtered signal from your USB-6009 DAQ device and displays the acquired signal on a waveform chart. The instructions are similar to Exercise 3.1, but you use DAQ Assistant in place of the Simulate Signal VI and you use Filter Express VI. Try to do this without following the instructions.

1. Open a blank VI.
2. Place a waveform chart on the front panel. Right-click to open the **Controls** palette and select **Controls»Modern»Graph»Waveform Chart**.
3. Place a numeric meter on the front panel. The meter is found in **Controls»Modern» Numeric»Meter**.
4. Right-click the y-axis on the waveform chart and deselect AutoScale Y.
5. Change the scale on the y-axis to –0.15 to 0.15 V by double-clicking the maximum and minimum axis values and typing the new value. Change the scale of the meter to 100 to 2000.
6. Go to the block diagram and place a While Loop around the chart and the meter (**Express»Execution Control**»**While Loop**).
7. Place the DAQ Assistant on the block diagram (**Express»Input**»**DAQ Assistant**). Choose analog input on channel ai0 of your device and click **Finish**. On the Task Timing tab, choose “Continuous” for the acquisition mode. If you are using the USB-6009, change the Input Range to – 2 to 2, the number of Samples to Read to 1000, and the Rate (Hz) to 44100.
8. Place the Filter Express VI to the right of the DAQ Assistant on the block diagram. From the **Functions** palette, select **Express»Signal Analysis**»**Filter** and place it on the block diagram inside the While Loop. In the configuration window under Filtering Type, choose “Highpass.” Under Cutoff Frequency, use a value of 300 Hz. Click **OK**.
9. Connect the “Data” output terminal of the DAQ Assistant to the “Signal” input of the Filter VI.
10. Connect the waveform chart to the “Filtered Signal” output.
11. Place a Tone Measurements Express VI on the block diagram (**Express»Signal Analysis**»**Tone**). Select “Frequency” under Single Tone Measurements**.**
12. Connect the output of the Filter VI to the “Signals” input of the Tone Measurements Express VI. Also, connect the “Frequency” output to the meter.
13. Return to the front panel and run the VI. Observe your acquired signal and its frequency and amplitude. Hum or whistle into the microphone and observe the frequency that you are producing.
14. Save the VI as `Exercise 3.2 - Data.vi`.
15. Close the VI.

Note: The solution to this exercise is printed in the back of this manual.

End of Exercise 3.2 (Track A)

Exercise 3.2 – Analysis (Track B)

Create a VI that measures the frequency of a filtered simulated signal and shows the difference between the filtered and unfiltered signal. You have to simulate the noise in the Simulate Signal Express VI. Try to do this without following the instructions.

1. Open a blank VI.
2. Place two waveform charts on the front panel. Right-click to open the **Controls** palette and select **Controls»Modern»Graph»Waveform Chart**.
3. Place a numeric meter on the front panel. The Meter is found in **Controls»Modern» Numeric»Meter**.
4. Right-click the y-axis on each waveform chart and deselect AutoScale Y.
5. Change the scale on the y-axis on both charts to –1 to 1 V by double-clicking the maximum and minimum axis values and typing the new value. Change the scale of the meter to 0 to 30.
6. Go to the block diagram and put a While Loop around the charts and the meters (**Express»Execution Control**»**While Loop**).
7. Place a Simulate Signal Express VI on the block diagram (**Express»Input**» **Simulate Sig**). Change the Frequency (Hz) to 20, Select to add noise, and set the Samples per Second (Hz) to 10000.
8. Ensure that the Noise Type is "Uniform White Noise" and the Noise Amplitude is "0.2."
9. Place the Filter Express VI to the right of the DAQ Assistant on the block diagram. From the **Functions** palette, select **Express»Signal Analysis**»**Filter** and place it on the block diagram inside the While Loop. In the configuration window under Filtering Type, choose "Lowpass." Under Cutoff Frequency, use a value of 35 Hz. Click **OK**.
10. Connect the output terminal of the Simulate Signal VI to the "Signal" input of the Filter VI. Connect the waveform charts to the "Filtered Signal" output and the output of the Simulate Signal VI.
11. Place a Tone Measurements Express VI on the block diagram (**Express»Signal Analysis**»**Tone**). Select "Frequency" under Single Tone Measurements**.**
12. Connect the output of the Filter VI to the "Signals" input of the Tone Measurements Express VI. Also, connect the "Frequency" output to the meter.
13. Return to the front panel and run the VI. Observe your acquired signal and its frequency and amplitude. Hum or whistle into the microphone and observe the frequency that you are producing.
14. Save the VI as `Exercise 3.2 - Data.vi`.
15. Close the VI.

Note: The solution to this exercise is printed in the back of this manual.

End of Exercise 3.2 (Track B)

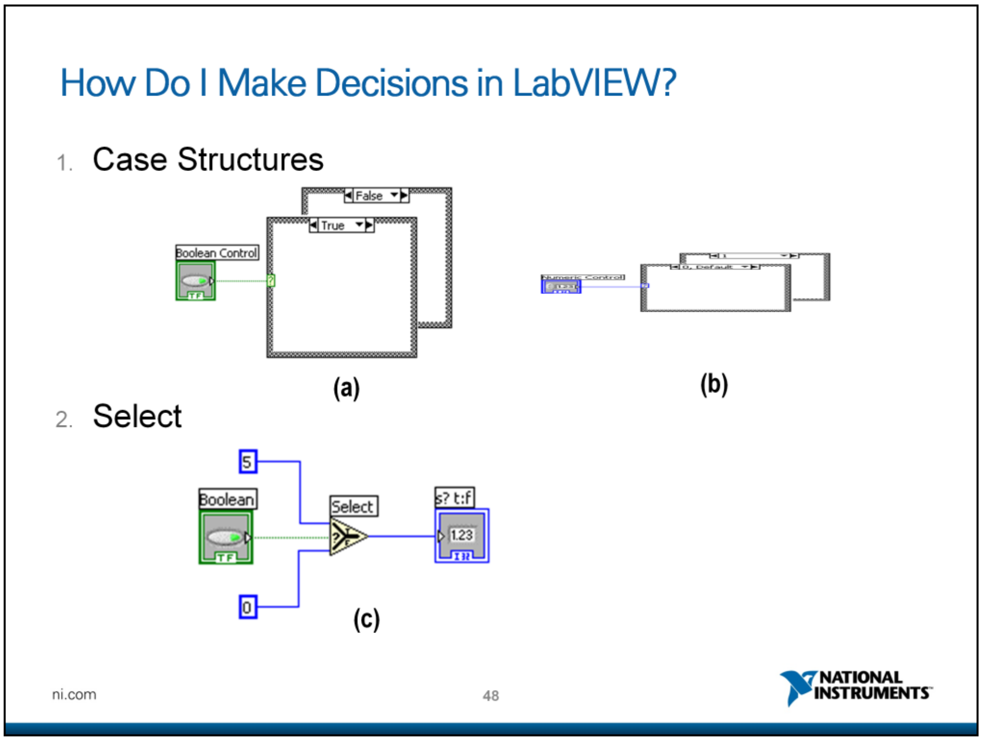
Exercise 3.2 – Analysis (Track C)

Create a VI that measures the frequency of the signal from your sound card and displays the acquired signal on a waveform chart. The instructions are similar to Exercise 3.1, but you use the Acquire Sound Signal VI in place of the Simulate Signal VI and a Filter Express VI. Try to do this without following the instructions.

1. Open a blank VI.
2. Place a waveform chart on the front panel. Right-click to open the **Controls** palette and select **Controls»Modern»Graph»Waveform Chart**.
3. Place a numeric meter on the front panel. The meter is found in **Controls» Numeric»Meter**.
4. Right-click the y-axis on the waveform chart and deselect AutoScale Y.
5. Change the scale on the y-axis to –1 to 1 V by double-clicking the maximum and minimum axis values and typing the new value. Change the scale of the meter to 100 to 2000.
6. Go to the block diagram and put a While Loop down (**Express»Execution Control»While Loop**).
7. Place the Acquire Sound Express VI on the block diagram (**Express»Input» Acquire Sound**).
8. Change the sample rate to 44100 and Click **OK**.
9. Place a Filter Express VI on the block diagram. In the configuration window, choose a highpass filter and a cutoff frequency of 300 Hz.
10. Place a Tone Measurements Express VI on the block diagram (**Express»Signal Analysis**»**Tone**). In the configuration window, choose “Amplitude” and “Frequency” measurements in the Single Tone Measurements section.
11. Connect the meter to the “Frequency” output of the Tone Measurements VI.
12. Connect the “Data” terminal of the Acquire Sound Express VI to the “Signal” input of the Filter VI.
13. Connect the “Filtered Signal” terminal of the Filter VI to the “Signals” input of the Tone Measurements VI. Select “Frequency” under Single Tone Measurements**.**
14. Return to the front panel and run the VI. Observe the signal from your sound card and its amplitude and frequency. Hum or whistle into the microphone and observe the amplitude and frequency you are producing.
15. Save the VI as `Exercise 3.2-Data.vi`. Close the VI.

Note: The solution to this exercise is printed in the back of this manual.

End of Exercise 3.2 (Track C)



How Do I Make Decisions in LabVIEW?

1. Case Structures

(a) (b)

2. Select

(c)

NATIONAL INSTRUMENTS

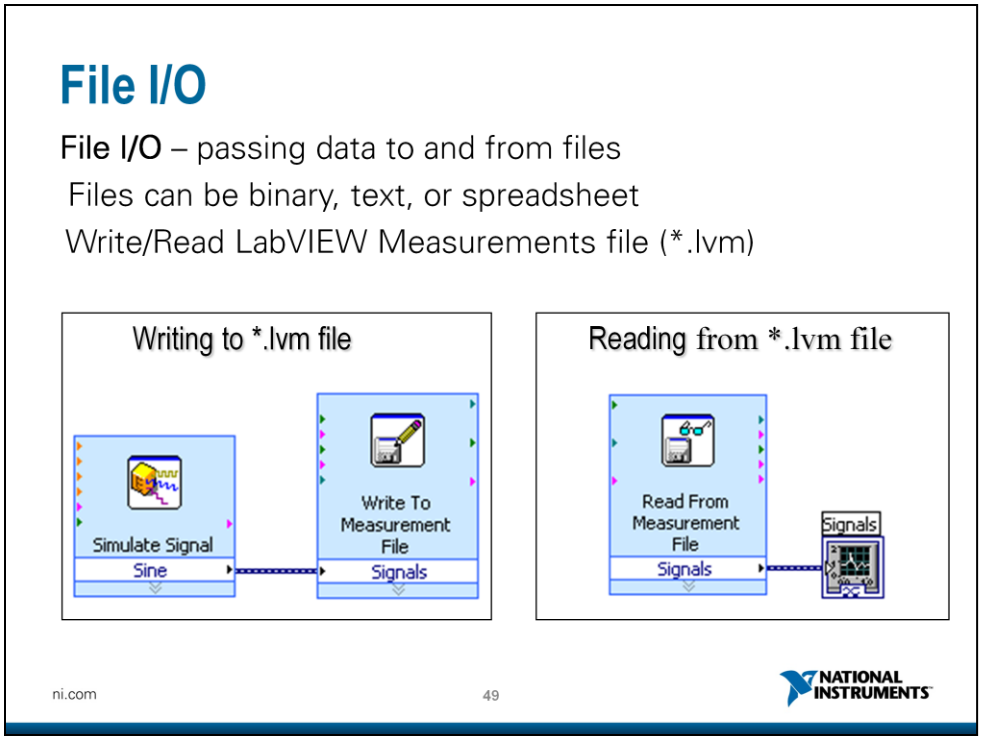
Case Structure

The case structure has one or more subdiagrams, or cases, one of which executes when the structure executes. The value wired to the selector terminal determines which case to execute and can be Boolean, string, integer, or enumerated type. Right-click the structure border to add or delete cases. Use the Labeling tool to enter value(s) in the case selector label and configure the value(s) handled by each case. It is found at **Functions» Programming»Structures»Case Structure**.

Select

Returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**. The connector pane displays the default data types for this polymorphic function. It is found at **Functions»Programming» Comparison»Select**.

- **Example A**: Boolean input. Simple if-then case. If the Boolean input is TRUE, the true case executes; otherwise the FALSE case executes.
- **Example B**: Numeric input. The input value determines which box to execute. If out of range of the cases, LabVIEW chooses the default case.
- **Example C**: When the Boolean passes a TRUE value to the Select VI, the value 5 is passed to the indicator. When the Boolean passes a FALSE value to the Select VI, 0 is passed to the indicator.

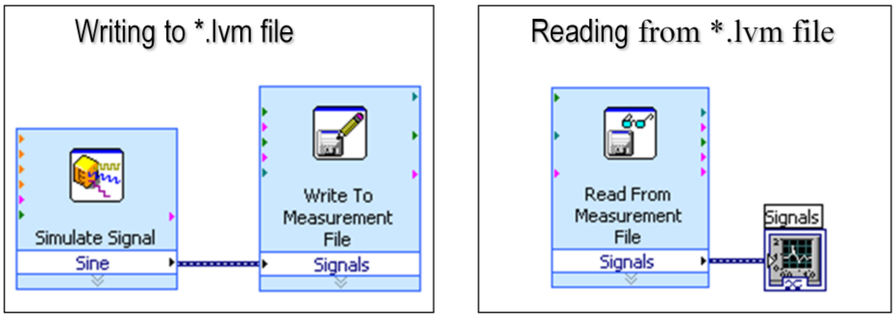


File I/O

File I/O – passing data to and from files

Files can be binary, text, or spreadsheet

Write/Read LabVIEW Measurements file (*.lvm)



ni.com 49 NATIONAL INSTRUMENTS

Use LabVIEW measurement data files to save data that the Write Measurement File Express VI generates. The LabVIEW data file is a tab-delimited text file you can open with a spreadsheet application or a text-editing application. In addition to the data an Express VI generates, the `.lvm` file includes information about the data, such as the date and time the data was generated.

File I/O operations pass data from memory to and from files. In LabVIEW, you can use File I/O functions to:

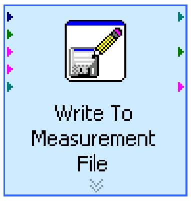
- Open and close data files
- Read data from and write data to files
- Read from and write to spreadsheet-formatted files
- Move and rename files and directories
- Change file characteristics
- Create, modify, and read a configuration file
- Write to or read from LabVIEW measurement files

In the next example, examine how to write to or read from LabVIEW measurement files (`*.lvm` files).

Exercise 3.3 – Decision Making and Saving Data (Tracks A, B, and C)

Create a VI that you can use to save your data to file if the frequency of your data goes below a user-controlled limit.

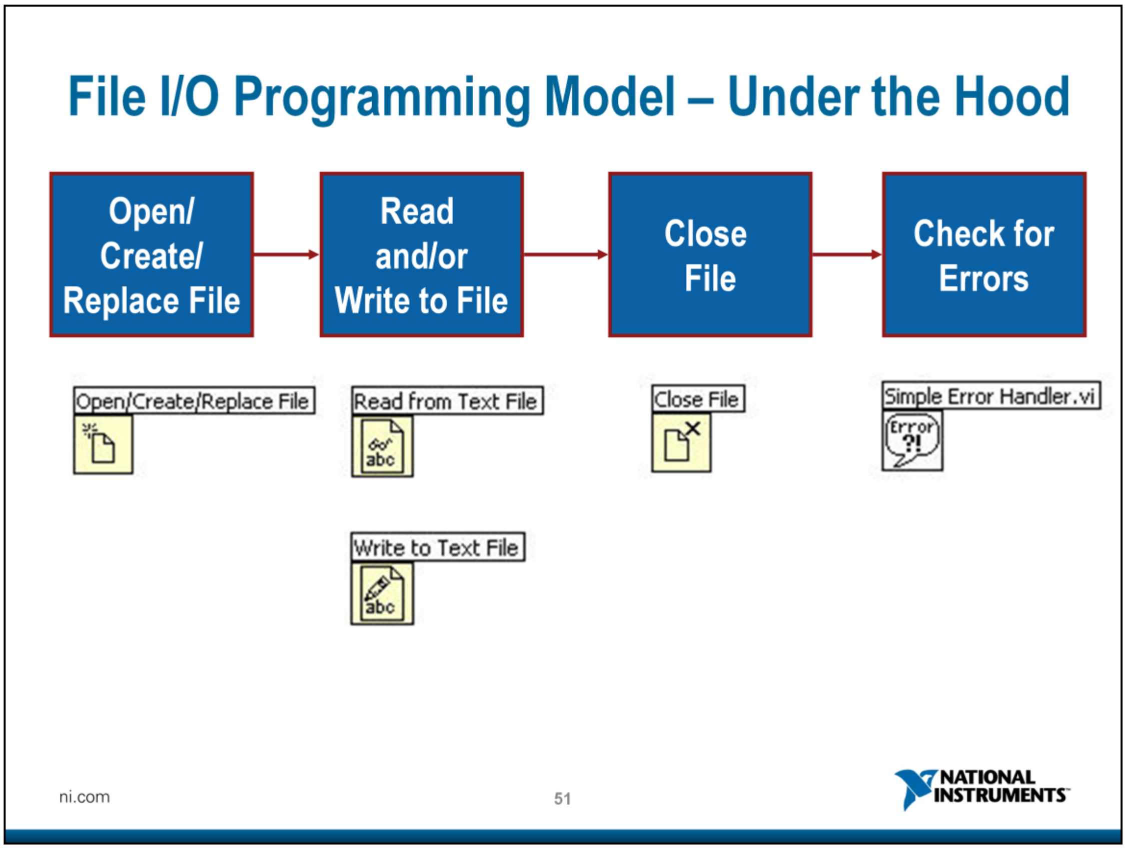
1. Open Exercise 3.2 – Data.vi.
2. Go to **File»Save As…** and save it as `Exercise 3.3 - Decision Making and Saving Data`. In the Save As dialog box, make sure **substitute copy for original** is selected and click **Continue…**
3. Add a case structure to the block diagram inside the While Loop (**Functions» Programming»Structures»Case Structure**).
4. Inside the "true" case of the case structure, add a Write to Measurement File Express VI (**Functions»Programming»File I/O»Write to Measurement File**).



   a. In the configuration window that opens, choose "Save to series of files (multiple files)." Note the default location your file will be saved to and change it if you wish.
   b. Click **Settings…** and choose "Use next available file name" under the **Existing Files** heading.
   c. Under **File Termination,** choose to start a new file after 10 segments. Click **OK** twice.
5. Add code so that if the frequency computed from the Tone Measurements Express VI goes below a user-controlled limit, the data will be saved to file. **Hint**: Go to **Functions»Programming»Comparison»Less?**
6. Remember to connect your data from the Filter Express VI to the "Signals" input of the Write to Measurement File VI. If you need help, refer to the solution of this exercise.
7. Go to the front panel and run your VI. Vary your frequency limit and then stop the VI.
8. Navigate to **My Documents»LabVIEW Data** (or the location you specified) and open one of the files that was saved there. Examine the file structure and check to verify that 10 segments are in the file.
9. Save your VI and close it.

Note: The solution to this exercise is printed in the back of this manual.

End of Exercise 3.3 (Tracks A, B, and C)



File I/O Programming Model – Under the Hood

Programming Model for the Intermediate File VIs

This same programming model applies to data acquisition, instrument control, file I/O, and most other communication schemes. In most instances, you open the file or communication channel, read and write multiple times, and then close or end the communication. It is also good programming practice to check for errors at the end. Remember this programming model when you move on to more advanced programming or look inside DAQ, communication, or File I/O Express VIs.

File I/O VIs and Functions

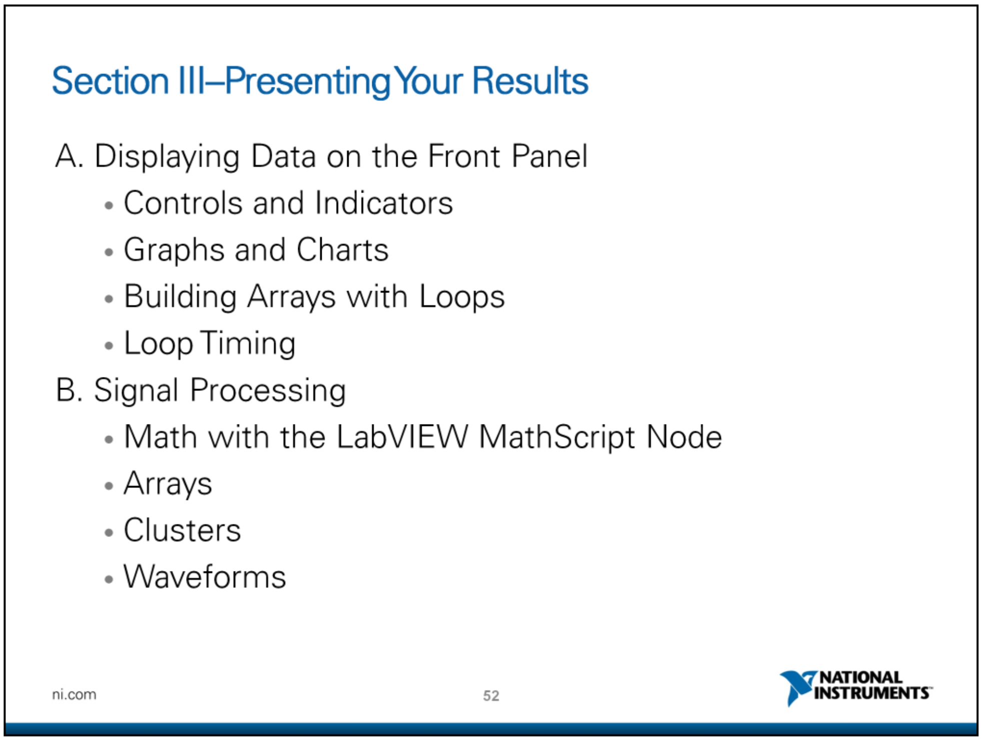
Use the File I/O VIs and functions to open and close files; read from and write to files; create directories and files you specify in the path control; retrieve directory information; and write strings, numbers, arrays, and clusters to files.

Use the high-level File I/O VIs located on the top row of the palette to perform common I/O operations, such as writing to or reading from various types of data. Acceptable types can include characters or lines in text files, 1D or 2D arrays of single-precision numeric values in spreadsheet text files, 1D or 2D arrays of single-precision numeric values in binary files, or 16-bit signed integers in binary files.

Use low-level File I/O VIs and functions located on the middle row of the palette as well as the Advanced File Functions to control each file I/O operation individually.

Use the principal low-level functions to create or open, write data to, read data from, and close a file. You also can use low-level functions to create directories; move, copy, or delete files; list directory contents; change file characteristics; or manipulate paths.

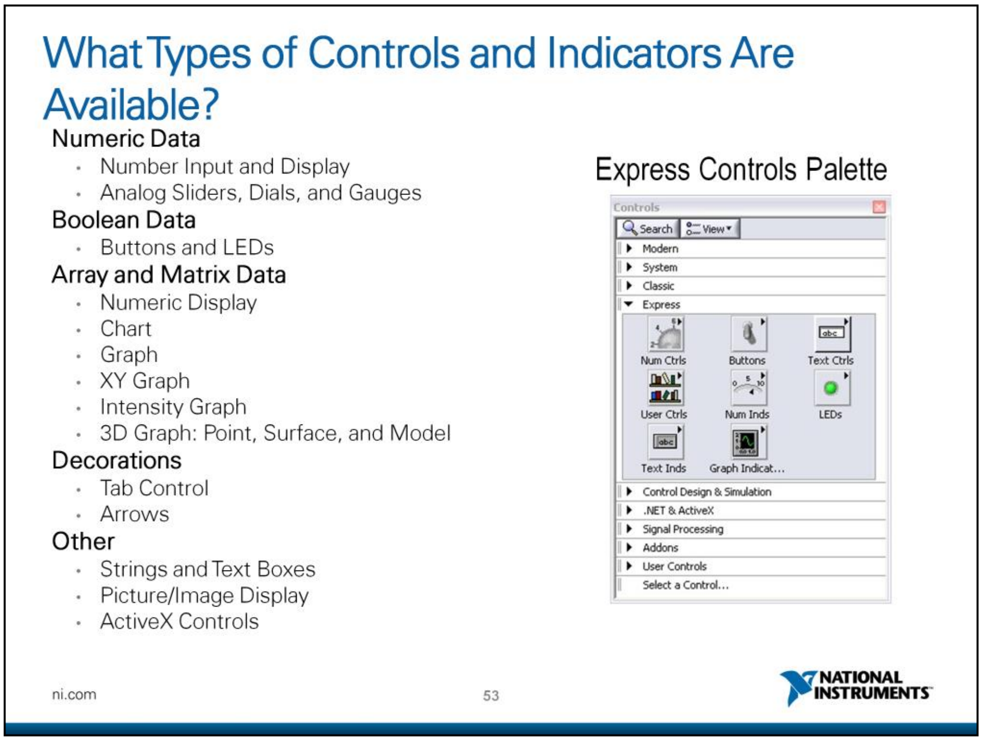
Refer to NI Developer Zone for more information about choosing a file format.



Section III–Presenting Your Results

A. Displaying Data on the Front Panel
- Controls and Indicators
- Graphs and Charts
- Building Arrays with Loops
- Loop Timing

B. Signal Processing
- Math with the LabVIEW MathScript Node
- Arrays
- Clusters
- Waveforms



What Types of Controls and Indicators Are Available?

Numeric Data
- Number Input and Display
- Analog Sliders, Dials, and Gauges

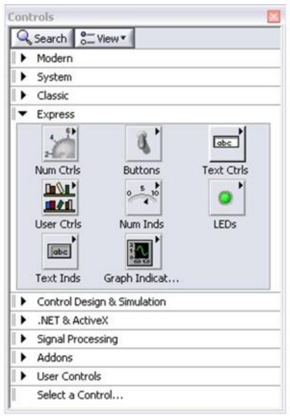
Boolean Data
- Buttons and LEDs

Array and Matrix Data
- Numeric Display
- Chart
- Graph
- XY Graph
- Intensity Graph
- 3D Graph: Point, Surface, and Model

Decorations
- Tab Control
- Arrows

Other
- Strings and Text Boxes
- Picture/Image Display
- ActiveX Controls

Express Controls Palette



ni.com 53

Controls and indicators are front panel items that you can use to interact with your program to provide input and display results. You can access controls and indicators by right-clicking the front panel.

In addition, you obtain additional controls and indicators when you install toolkits and modules.

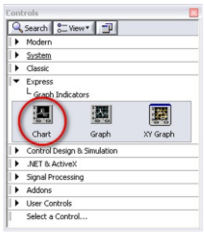
For example, when you install the control design tools, you can use specialized plots, such as Bode and Nyquist, that are not available by default.

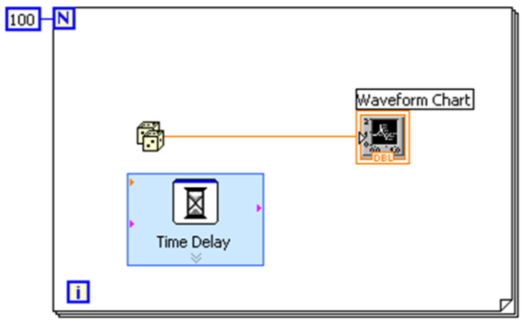
Charts – Add 1 Data Point at a Time with History

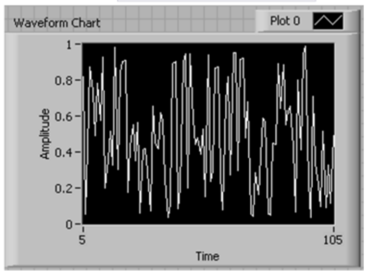
Waveform chart – special numeric indicator that can display a history of values

Chart updates with each individual point it receives

Controls»Express»Graph Indicators»Chart

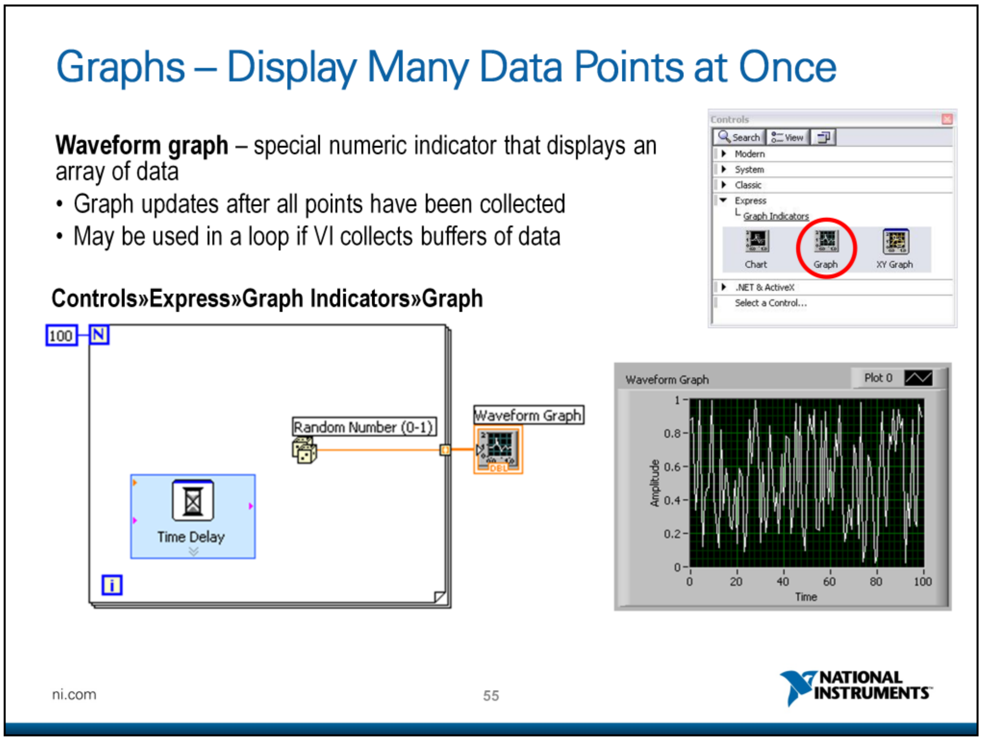






The waveform chart is a special numeric indicator that displays one or more plots. It is located on the **Controls»Modern»Graph** palette. Waveform charts can display single or multiple plots. The following front panel shows an example of a multiplot waveform chart.

You can change the minimum and maximum values of either the x-axis or y-axis by double-clicking on the value with the labeling tool and typing the new value. Similarly, you can change the label of the axis. You can also right-click the plot legend and change the style, shape, and color of the trace that is displayed on the chart.



Graphs – Display Many Data Points at Once

Waveform graph – special numeric indicator that displays an array of data

- Graph updates after all points have been collected
- May be used in a loop if VI collects buffers of data

Controls»Express»Graph Indicators»Graph

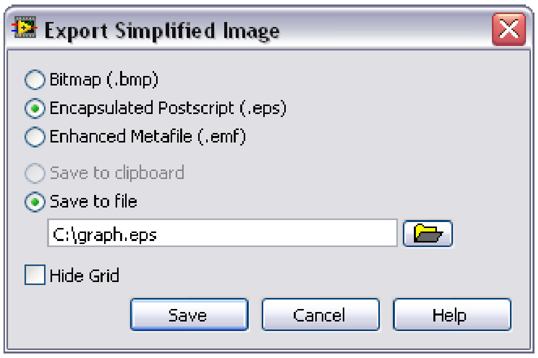
Controls
Search
View
Modern
System
Classic
Express
Graph Indicators
Chart
Graph
XY Graph
.NET & ActiveX
Select a Control...
100
N
Random Number (0-1)
Waveform Graph
DBL
Time Delay
i
Waveform Graph
Plot 0
Amplitude
Time

ni.com 55 NATIONAL INSTRUMENTS

Graphs are very powerful indicators in LabVIEW. You can use these highly customizable tools to concisely display a great deal of information.

With the properties page of the graph, you can display settings for plot types, scale and cursor options, and many other features of the graph. To open the properties page, right-click the graph on the front panel and choose **Properties.**

You can also create technical-paper-quality graphics with the "export simplified image" function. Right-click the graph and select **Data Operations»Export Simplified Image…**



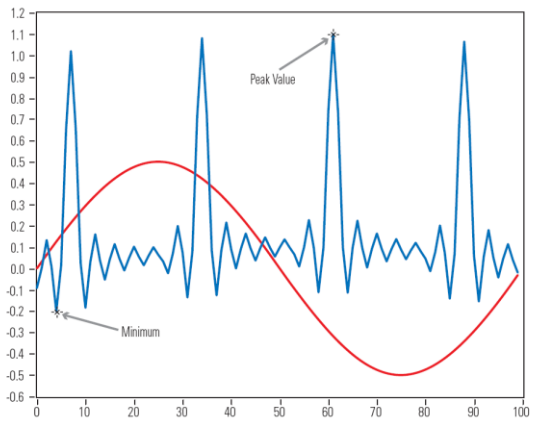
Export Simplified Image

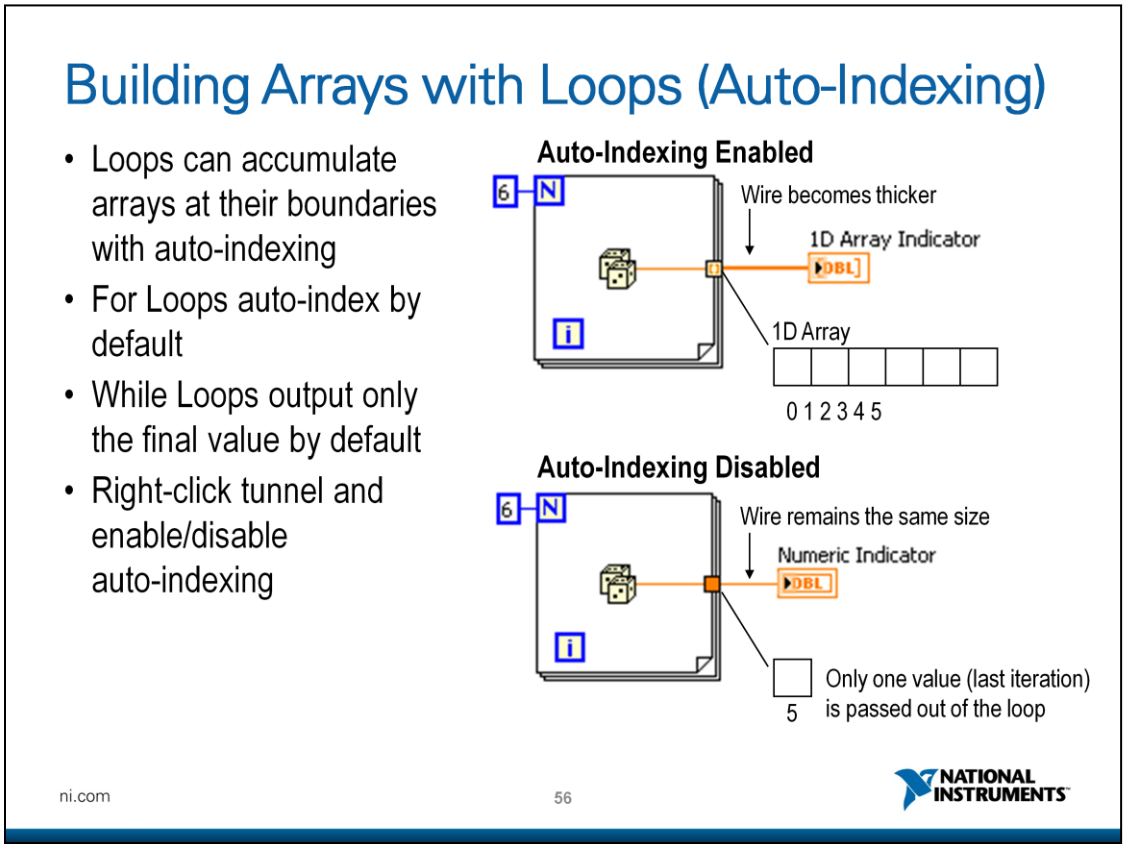
Bitmap (.bmp) [✓ Encapsulated Postscript (.eps)] Enhanced Metafile (.emf)

Save to clipboard [✓ Save to file]

C:\graph.eps

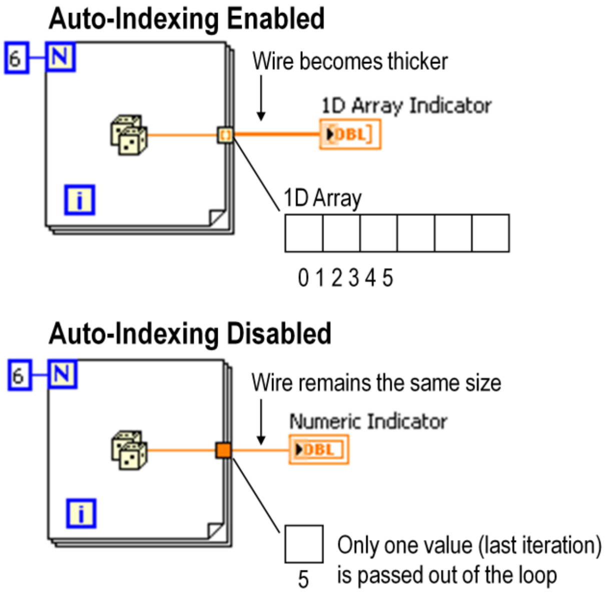
Hide Grid





Building Arrays with Loops (Auto-Indexing)

- Loops can accumulate arrays at their boundaries with auto-indexing
- For Loops auto-index by default
- While Loops output only the final value by default
- Right-click tunnel and enable/disable auto-indexing

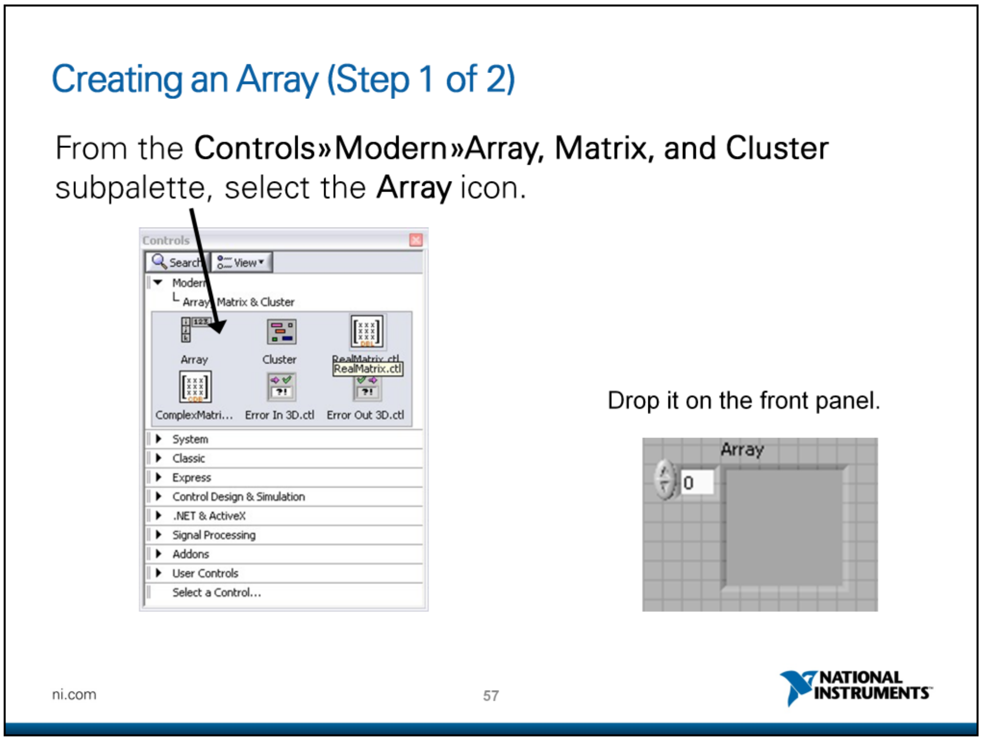


For Loops and While Loops can index and accumulate arrays at their boundaries. This is known as auto-indexing.

- The indexing point on the boundary is called a tunnel
- The For Loop is auto-indexing-enabled by default
- The While Loop is auto-indexing-disabled by default

Examples:

- Enable auto-indexing to collect values within the loop and build the array. All values are placed in the array upon exiting the loop.
- Disable auto-indexing if you are interested only in the final value.



Creating an Array (Step 1 of 2)

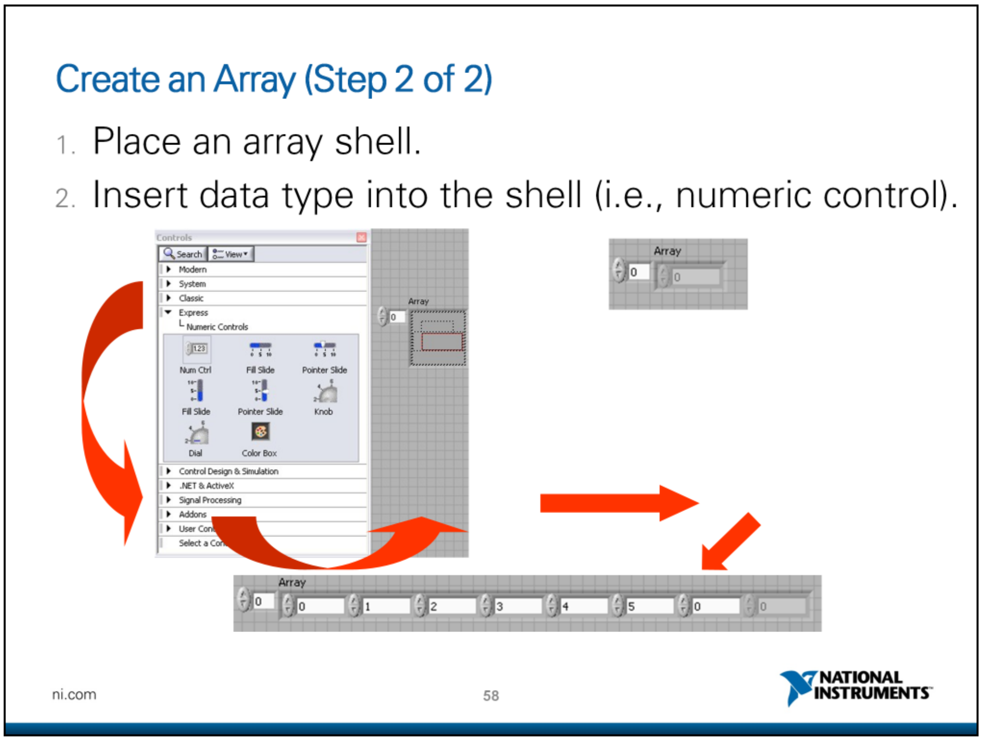
From the **Controls»Modern»Array, Matrix, and Cluster** subpalette, select the **Array** icon.

Drop it on the front panel.

ni.com 57 NATIONAL INSTRUMENTS

To create an array control or indicator as shown, select an array on the **Controls»Modern» Array, Matrix, and Cluster** palette, place it on the front panel, and drag a control or indicator into the array shell. If you attempt to drag an invalid control or indicator such as an XY graph into the array shell, you are unable to drop the control or indicator in the array shell.

You must insert an object in the array shell before you use the array on the block diagram. Otherwise, the array terminal appears black with an empty bracket.



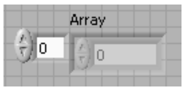
Create an Array (Step 2 of 2)

1. Place an array shell.
2. Insert data type into the shell (i.e., numeric control).

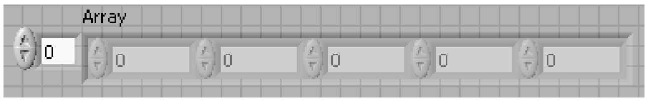
ni.com 58 NATIONAL INSTRUMENTS

To add dimensions to an array one at a time, right-click the index display and select **Add Dimension** from the shortcut menu. You also can use the Positioning tool to resize the index display until you have as many dimensions as you want.

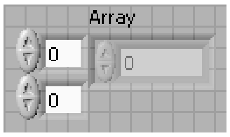
1D Array Viewing a Single Element



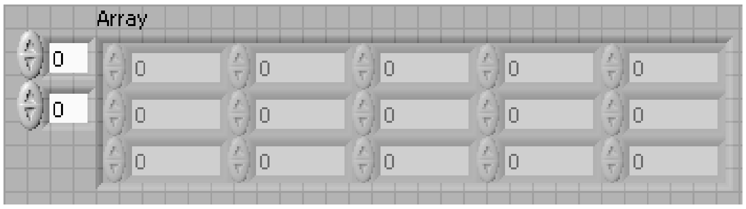
1D Array Viewing Multiple Elements

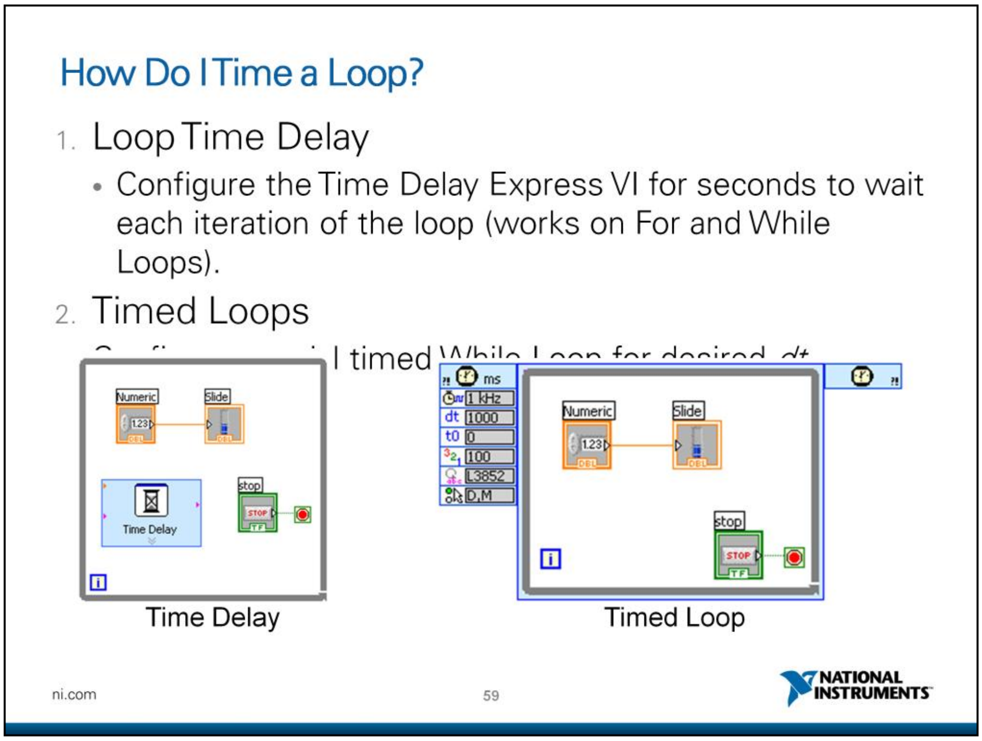


2D Array Viewing a Single Element



2D Array Viewing Multiple Elements





How Do I Time a Loop?

1. Loop Time Delay
   - Configure the Time Delay Express VI for seconds to wait each iteration of the loop (works on For and While Loops).
2. Timed Loops
   - timed While Loop for desired *dt*

Time Delay

Timed Loop

NATIONAL INSTRUMENTS

Time Delay

The Time Delay Express VI delays execution by a specified number of seconds. Following the rules of dataflow programming, the While Loop does not iterate until all tasks inside of it are complete, thus delaying each iteration of the loop.

Timed Loops

Executes each iteration of the loop at the period you specify. Use the timed loop when you want to develop VIs with multirate timing capabilities, precise timing, feedback on loop execution, timing characteristics that change dynamically, or several levels of execution priority.

Double-click the Input Node or right-click the Input Node and select **Configure Timed Loop** from the shortcut menu to display the Loop Configuration dialog box, where you can configure the timed loop. The values you enter in the **Loop Configuration** dialog box appear as options in the Input Node.

Wait Until Next ms Multiple

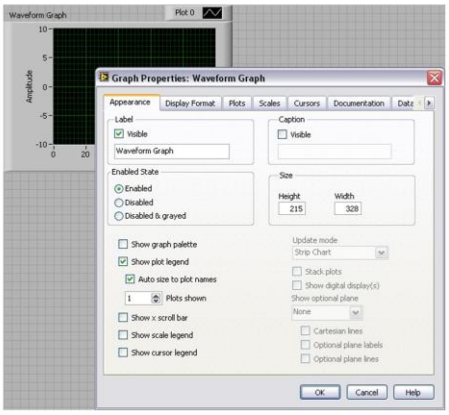
This function waits until the value of the millisecond timer becomes a multiple of the specified **millisecond multiple** to help you synchronize activities. You can call this function in a loop to control the loop execution rate. However, it is possible that the first loop period might be short. This function makes asynchronous system calls, but the nodes themselves function synchronously. Therefore, it does not complete execution until the specified time has elapsed. This function can be found at **Functions» Programming»Timing»Wait Until Next ms Multiple**.

Control and Indicator Properties

Properties are characteristics or qualities about an object

Properties can be found by right-clicking on a control or indicator

- Properties include:
  - Size
  - Color
  - Plot style
  - Plot color
- Features include:
  - Cursors
  - Scaling



Graph Properties: Waveform Graph

Appearance | Display Format | Plots | Scales | Cursors | Documentation | Data

Label: [✓ Visible] Waveform Graph

Caption: [ ] Visible

Enabled State: [✓ Enabled] Disabled / Disabled & grayed

Size: Height 215, Width 328

[ ] Show graph palette
[✓] Show plot legend
[✓] Auto size to plot names
1 Plots shown
[ ] Show x scroll bar
[ ] Show scale legend
[ ] Show cursor legend

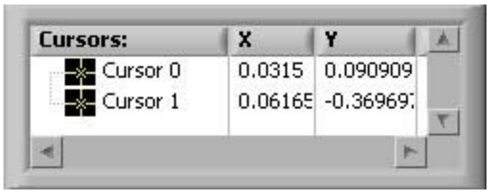
Update mode: Strip Chart
[ ] Stack plots
[ ] Show digital display(s)
Show optional plane: None
[ ] Cartesian lines
[ ] Optional plane labels
[ ] Optional plane lines

Properties are all the qualities of a front panel object. With properties, you can set or read such characteristics as foreground and background color, data formatting and precision, visibility, descriptive text, size and location on the front panel, and so on.

Exercise 4 – Manual Analysis (Tracks A, B, and C)

Create a VI that displays simulated data on a waveform graph and measures the frequency and amplitude of that data. Use cursors on the graph to verify the frequency and amplitude measurements.

1. Open Exercise 3.1 – Simulated.vi.
2. Save the VI as `Exercise 4 – Manual Analysis.vi`.
3. Go to the block diagram and remove the While Loop. Right-click the edge of the loop and choose **Remove While Loop** so that the code inside the loop does not get deleted.
4. Delete the **Stop** control.
5. On the front panel, replace the waveform chart with a waveform graph. Right-click the chart and select **Replace»Modern»Graph»Waveform Graph**.
6. Make the cursor legend viewable on the graph. Right-click on the graph and select **Visible Items**»**Cursor Legend**.
7. Change the maximum value of the "Frequency In" dial to 100. Double-click on the maximum value and type `100` once the text is highlighted.
8. Set a default value for the "Frequency In" dial by setting the dial to the value you would like, right-clicking the dial, and selecting **Data Operations»Make Current Value Default**.
9. Run the VI and observe the signal on the waveform graph. If you cannot see the signal, you may need to turn on auto-scaling for the x-axis. Right-click on the graph and select **X Scale»AutoScale X**.
10. Change the frequency of the signal. Run the VI again so you can see a few periods on the graph.
11. Manually measure the frequency and amplitude of the signal on the graph using cursors. Right-click on the graph and select **Visible Items »Cursor Legend** to show the cursor legend. To add a cursor on the graph, right-click on the cursor pane and choose **Create Cursor»Free.** Once the cursors are displayed, you can drag them around on the graph and their coordinates appear in the cursor legend.

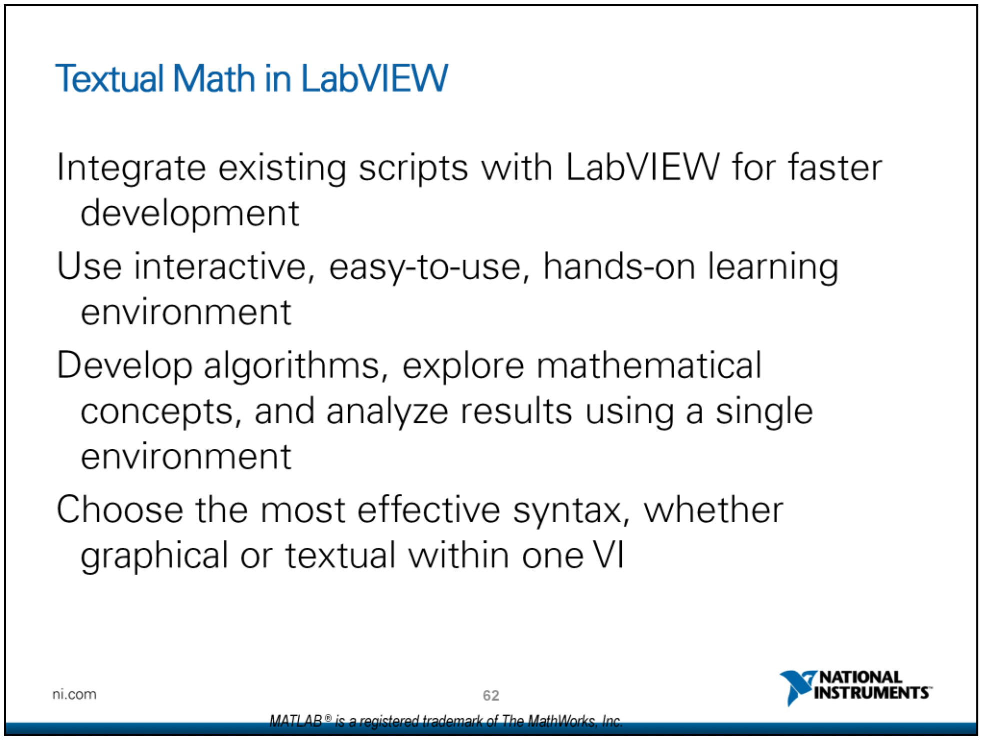


Cursors:	X	Y
Cursor 0	0.0315	0.090909
Cursor 1	0.06165	-0.36969

12. Remember that the frequency of a signal is the reciprocal of its period (f = 1/T). Does your measurement match the frequency and amplitude indicators from the Tone Measurements VI?
13. Save your VI and close it.

Note: The solution to this exercise is printed in the back of this manual.

End of Exercise 4 (Tracks A, B, and C)



Textual Math in LabVIEW

Integrate existing scripts with LabVIEW for faster development

Use interactive, easy-to-use, hands-on learning environment

Develop algorithms, explore mathematical concepts, and analyze results using a single environment

Choose the most effective syntax, whether graphical or textual within one VI

ni.com 62

NATIONAL INSTRUMENTS

MATLAB® is a registered trademark of The MathWorks, Inc.

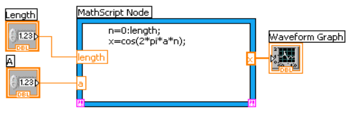
Overview

Asof LabVIEW 8.6, you have new freedom to choose the most effective syntax for technical computing, whether you are developing algorithms, exploring DSP concepts, or analyzing results. You can instrument your scripts and develop algorithms on the block diagram by interacting with popular third-party math tools such as The MathWorks, Inc. MATLAB® software, Wolfram Mathematica, Maplesoft Maple, MathSoft Mathcad, ITT IDL, and NI Xmath. Use of these math tools with LabVIEW is achieved in a variety of ways depending on the vendor as listed below:

Native LabVIEW textual math node

LabVIEW MathScript Node, Formula Node

Communication with vendor software through LabVIEW node



```
n=0:length;
x=cos(2*pi*a*n);
```

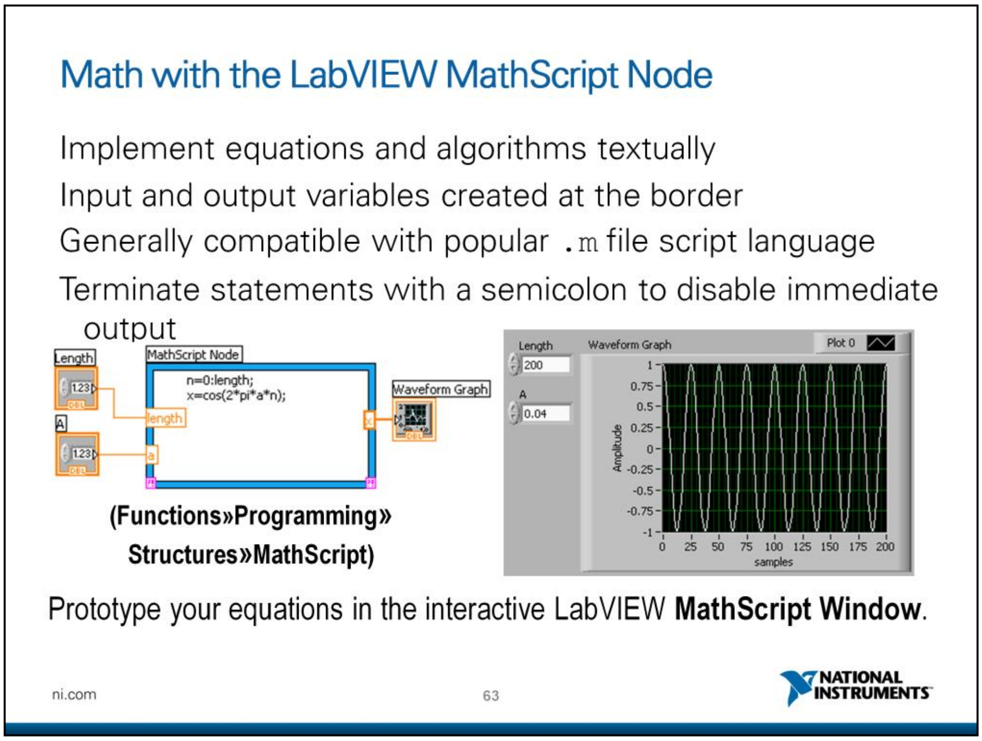
Xmath node, MATLAB script Node, Maple* Node, IDL* Node

Communication with vendor software through VI Server

Mathematica* VIs, Mathcad* VIs

In LabVIEW, you can combine the intuitive LabVIEW graphical dataflow programming with LabVIEW MathScript, a math-oriented textual programming language that is generally compatible with popular `.m` file script language.

*LabVIEW module specific to the math tool must be installed.



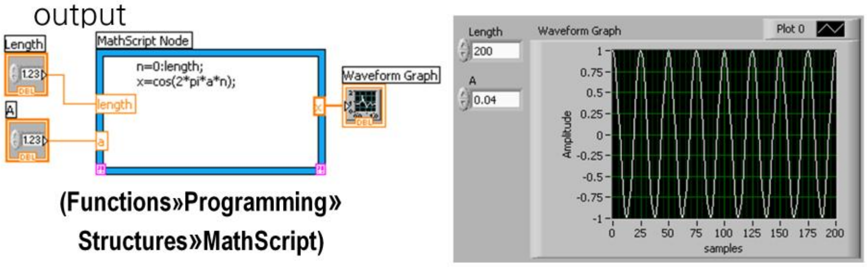
Math with the LabVIEW MathScript Node

Implement equations and algorithms textually

Input and output variables created at the border

Generally compatible with popular `.m` file script language

Terminate statements with a semicolon to disable immediate output



```
n=0:length;
x=cos(2*pi*a*n);
```

(Functions»Programming» Structures»MathScript)

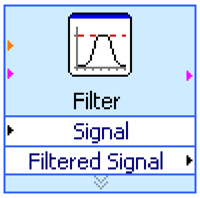
Prototype your equations in the interactive LabVIEW **MathScript Window**.

The LabVIEW MathScript Node enhances LabVIEW by adding a native text-based language for mathematical algorithm implementation in the graphical programming environment. You can open and use the `.m` file scripts you've written and saved from the LabVIEW MathScript window in the LabVIEW MathScript Node. The `.m` file scripts you created in other math software generally run as well. With LabVIEW MathScript you can pick the syntax you are most comfortable with to solve the problem. You can instrument equations with the LabVIEW MathScript Node for parameter exploration, simulation, or deployment in a final application.

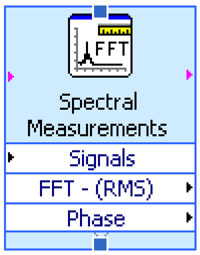
Exercise 5 – Apply What You Have Learned (Tracks A, B, and C)

In this exercise, create a VI that uses what you have learned. Design a VI that does the following:

1. Acquires data from your device (or creates a simulated signal) and graphs it.
2. Filters that data using the Filter Express VI (**Functions»Express»Signal Analysis» Filter**). There should be a front panel control for a user-configurable cutoff frequency.

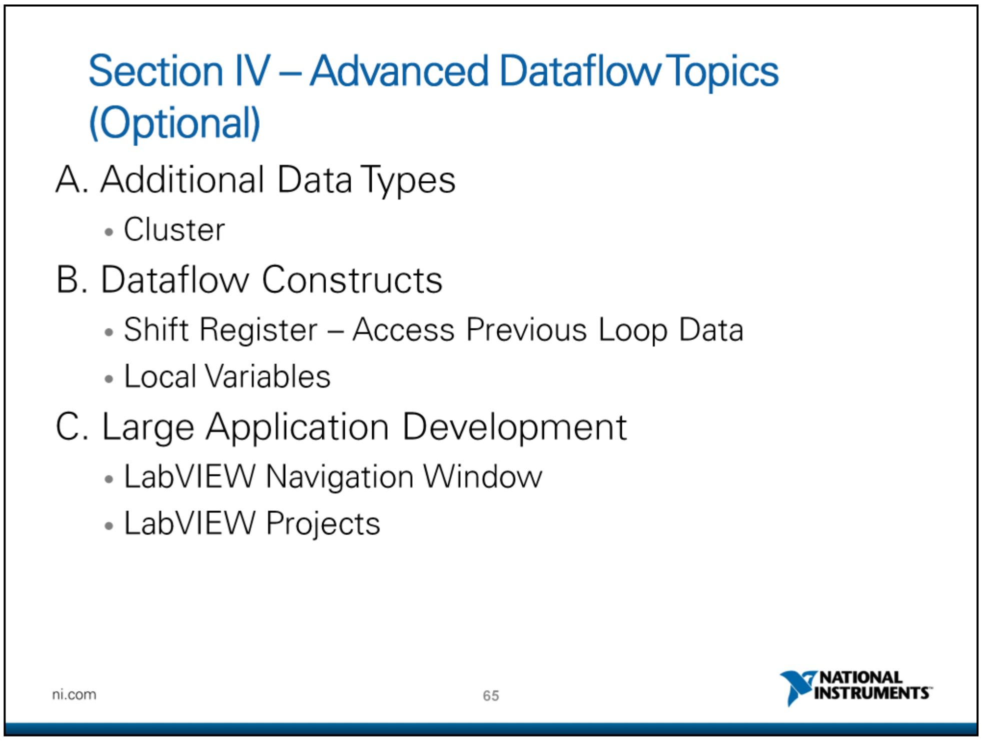


3. Takes a fast Fourier transform to get the frequency information from the filtered data and graph the result. Use the Spectral Measurements Express VI (**Functions» Express»Signal Analysis»Spectral**).



4. Finds the dominant frequency of the filtered data using the Tone Measurements Express VI.
5. Compares that frequency to a user-inputted limit. If the frequency is over that limit, light up an LED on the front panel. If you have a USB-6009, light up the LED on your hardware using the DAQ Assistant. You need to invert the digital line for the LED to light up when over the limit. You can specify this in the configuration window of the DAQ Assistant or with a “not” Boolean function.
6. If you get stuck, open up the solution or view it at the end of this manual.

End of Exercise 5 (Tracks A, B, and C)

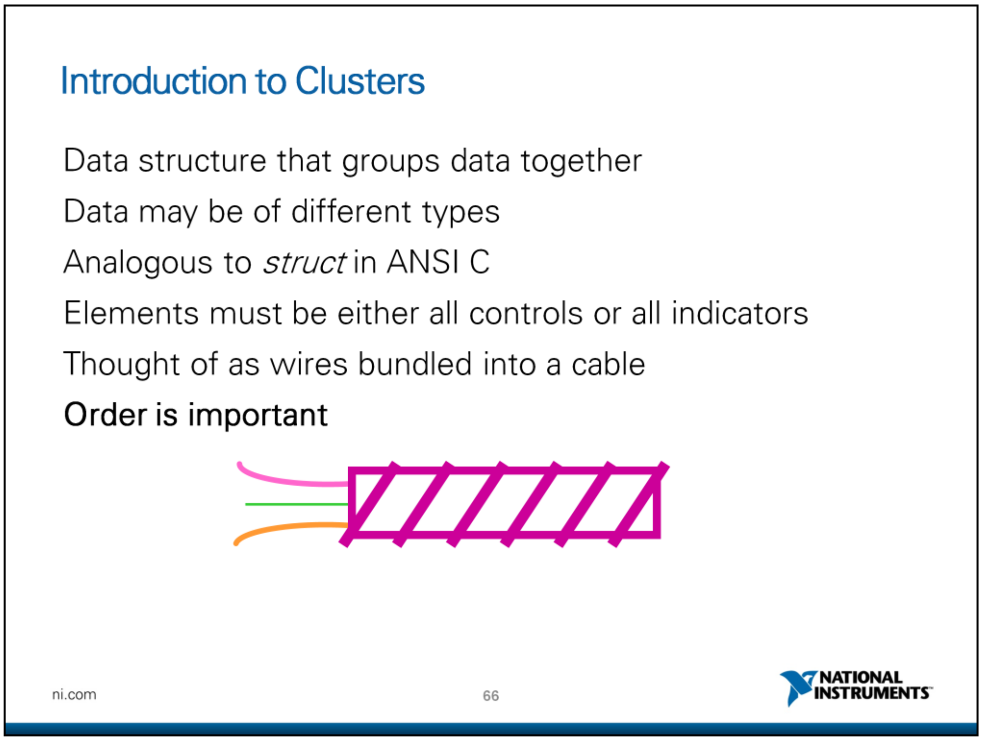


Section IV – Advanced Dataflow Topics (Optional)

A. Additional Data Types
- Cluster

B. Dataflow Constructs
- Shift Register – Access Previous Loop Data
- Local Variables

C. Large Application Development
- LabVIEW Navigation Window
- LabVIEW Projects



Introduction to Clusters

Data structure that groups data together

Data may be of different types

Analogous to *struct* in ANSI C

Elements must be either all controls or all indicators

Thought of as wires bundled into a cable

Order is important

ni.com 66 NATIONAL INSTRUMENTS

Clusters group like or unlike components together. They are equivalent to a *record* in Pascal or a *struct* in ANSI C.

Cluster components may be of different data types.

Examples

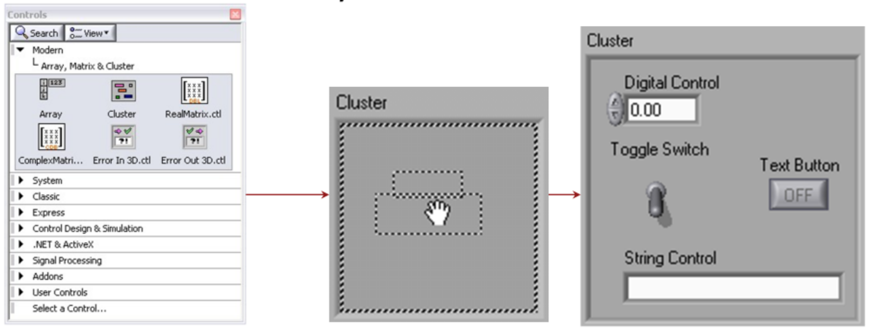
- Error information – Grouping a Boolean error flag, a numeric error code, and an error source string to specify the exact error.
- User information – Grouping a string indicating a user's name and an ID number specifying the user's security code.

All elements of a cluster must be either controls or indicators. You cannot have a string control and a Boolean indicator. Think of clusters as grouping individual wires (data objects) together into a cable (cluster).

Creating a Cluster

1. Select a **Cluster** shell.
2. Place objects inside the shell.

Controls»Modern»Array, Matrix & Cluster



Create a cluster front panel object by choosing **Cluster** from the **Controls»Modern» Array, Matrix & Cluster** palette.

- This option gives you a shell (similar to the array shell when creating arrays).
- You can size the cluster shell when you drop it.
- Right-click inside the shell and add objects of any type.

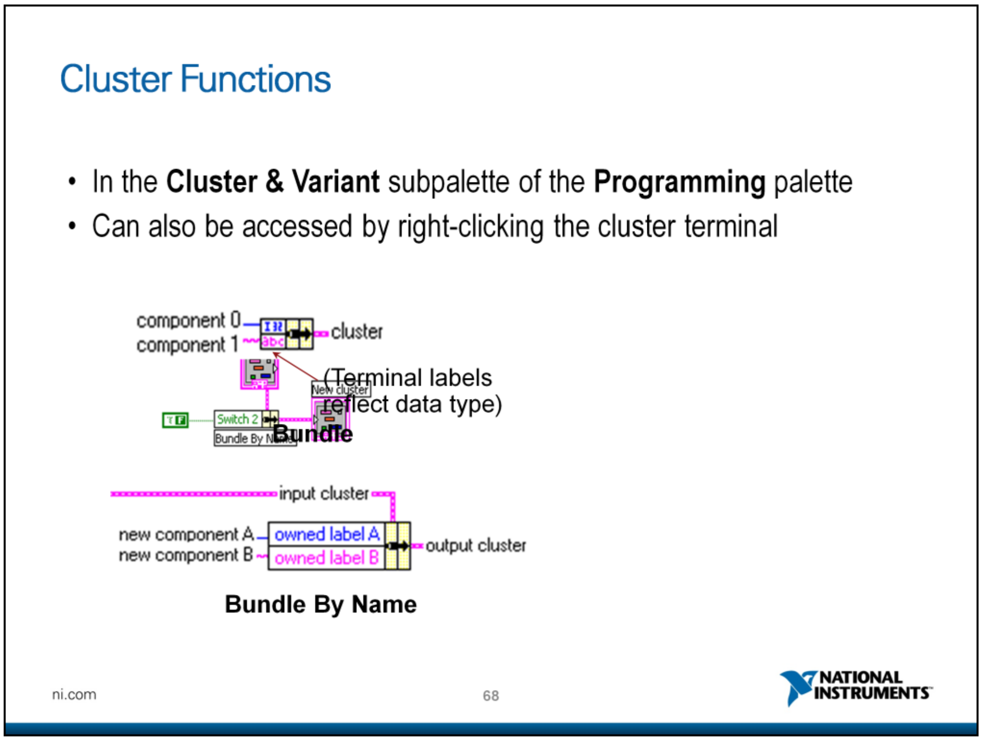
Note: You can even have a cluster inside of a cluster.

The cluster becomes a control or an indicator cluster based on the first object you place inside the cluster.

You can also create a cluster constant on the block diagram by choosing **Cluster Constant** from the **Cluster** palette.

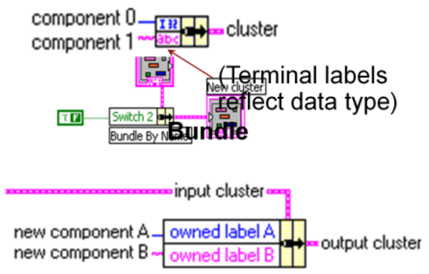
- This gives you an empty cluster shell.
- You can size the cluster when you drop it.
- Put other constants inside the shell.

Note: You cannot place terminals for front panel objects in a cluster constant on the block diagram, nor can you place "special" constants like the Tab or Empty String constant within a block diagram cluster shell.



Cluster Functions

- In the **Cluster & Variant** subpalette of the **Programming** palette
- Can also be accessed by right-clicking the cluster terminal



Bundle By Name

ni.com 68 NATIONAL INSTRUMENTS

The terms bundle and cluster are closely related in LabVIEW.

Example: You use a bundle function to create a cluster. You use an unbundle function to extract the parts of a cluster.

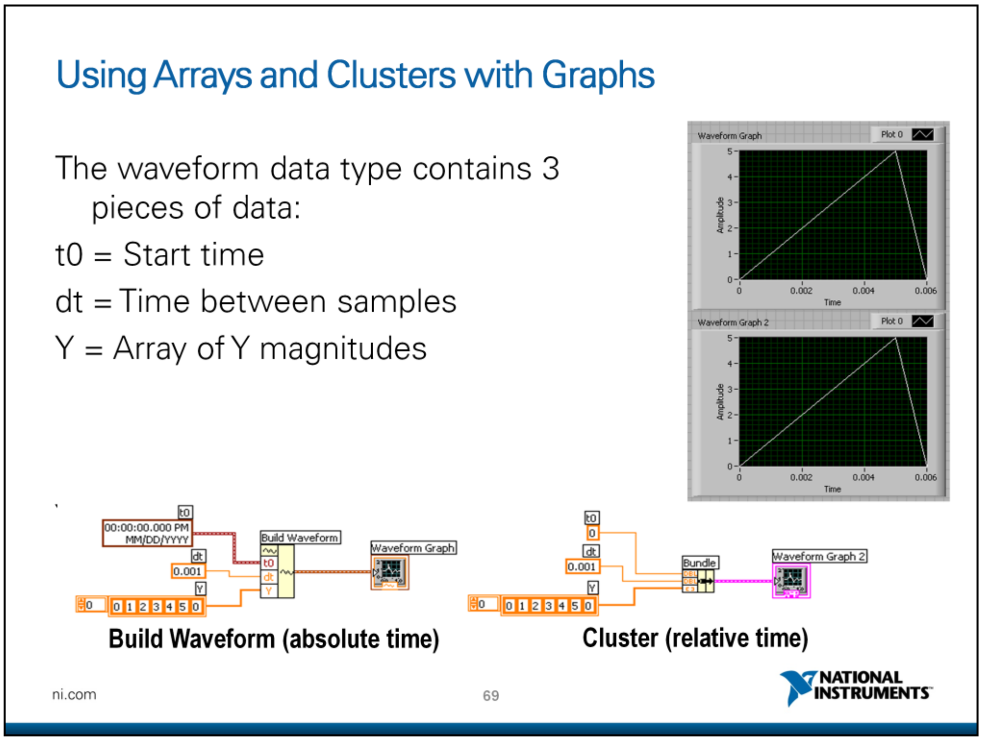
Bundle – Forms a cluster containing the given objects in the specified order.

Bundle by Name – Updates specific cluster object values (the object must have an owned label).

Unbundle – Splits a cluster into each of its individual elements by data type.

Unbundle by Name – Returns the cluster elements whose names you specify.

Note: You must have an existing cluster wired into the middle terminal of the function to use Bundle By Name.



Using Arrays and Clusters with Graphs

The waveform data type contains 3 pieces of data:

t0 = Start time

dt = Time between samples

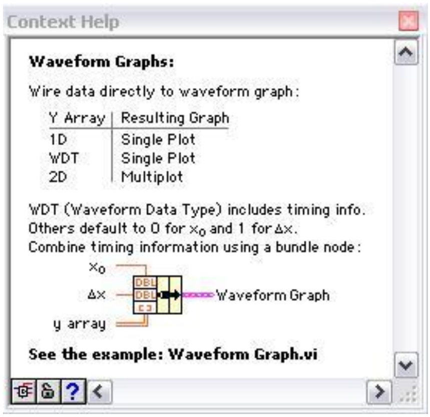
Y = Array of Y magnitudes

Waveform Graph
Plot 0
Amplitude
Time
Waveform Graph 2
t0
00:00:00.000 PM MM/DD/YYYY
dt
0.001
Y
Build Waveform
0
Bundle

Build Waveform (absolute time)

Cluster (relative time)

The waveform data type carries the data, start time, and Δ*t* of a waveform. You can create waveforms using the Build Waveform function. Many of the VIs and functions you use to acquire or analyze waveforms accept and return the waveform data type by default. When you wire a waveform data type to a waveform graph or chart, the graph or chart automatically plots a waveform based on the data, start time, and Δ*x* of the waveform. When you wire an array of waveform data types to a waveform graph or chart, the graph or chart automatically plots all the waveforms.



Context Help

Waveform Graphs:

Wire data directly to waveform graph:

Y Array	Resulting Graph
1D	Single Plot
WDT	Single Plot
2D	Multiplot

WDT (Waveform Data Type) includes timing info. Others default to 0 for x_0 and 1 for Δx. Combine timing information using a bundle node:

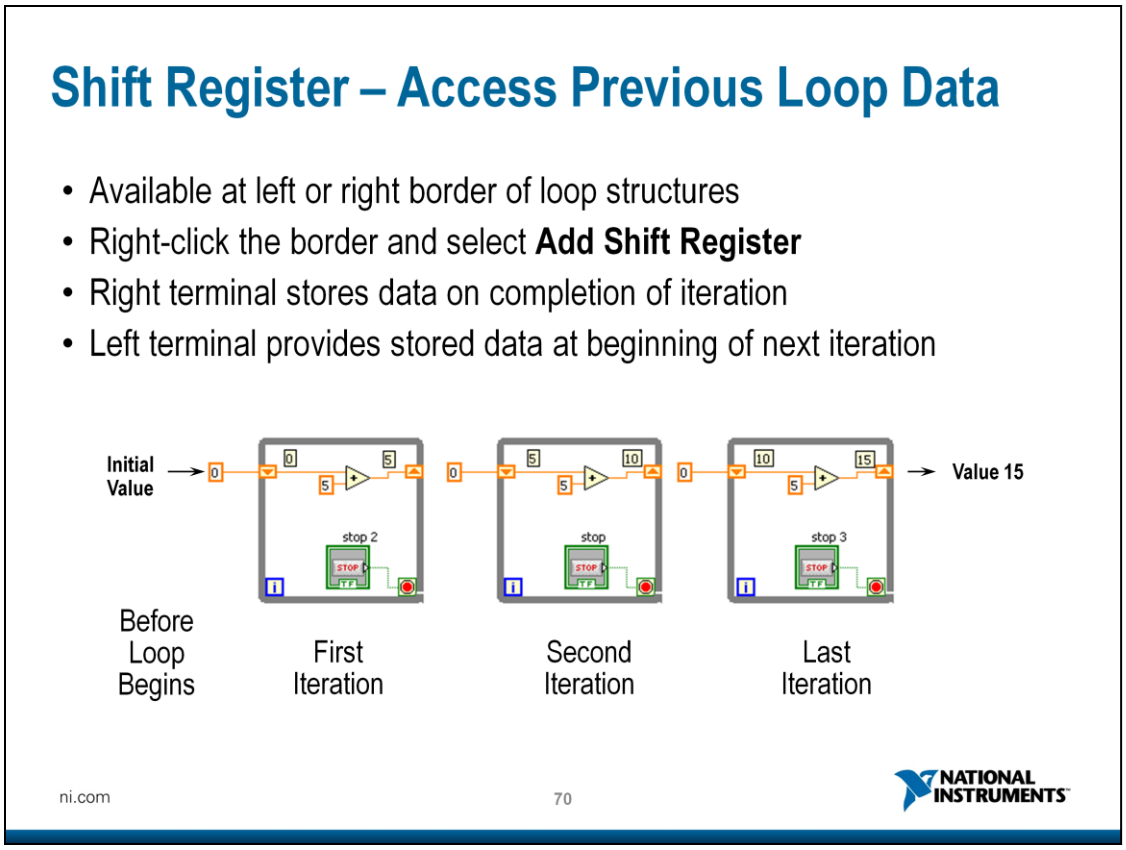
See the example: Waveform Graph.vi

Build Waveform

Builds a waveform or modifies an existing waveform with the start time represented as an absolute timestamp. Timestamps are accurate to real-world time and date and are very useful for real-world data recording.

Bundle

Builds a waveform or modifies an existing waveform with a relative timestamp. The input to t_0 is a DBL. By building waveforms using the bundle, you can plot data on the negative x-axis (time).



Shift Register – Access Previous Loop Data

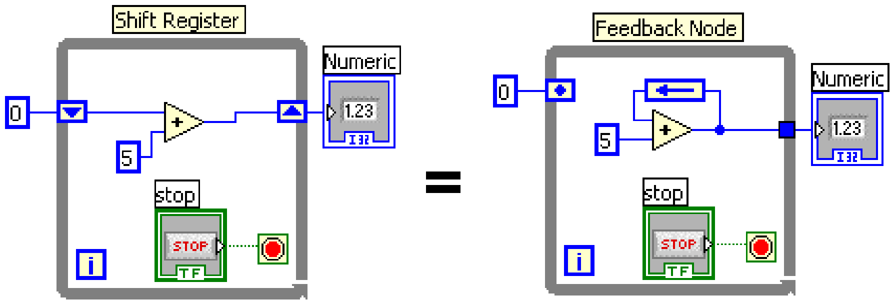
- Available at left or right border of loop structures
- Right-click the border and select **Add Shift Register**
- Right terminal stores data on completion of iteration
- Left terminal provides stored data at beginning of next iteration

ni.com 70 NATIONAL INSTRUMENTS

Shift registers transfer data from one iteration to the next:

- Right-click on the left or right side of a For Loop or a While Loop and select **Add Shift Register**.
- The right terminal stores data at the end of an iteration. Data appears at the left terminal at the start of the next iteration.
- A shift register adapts to any data type wired into it.

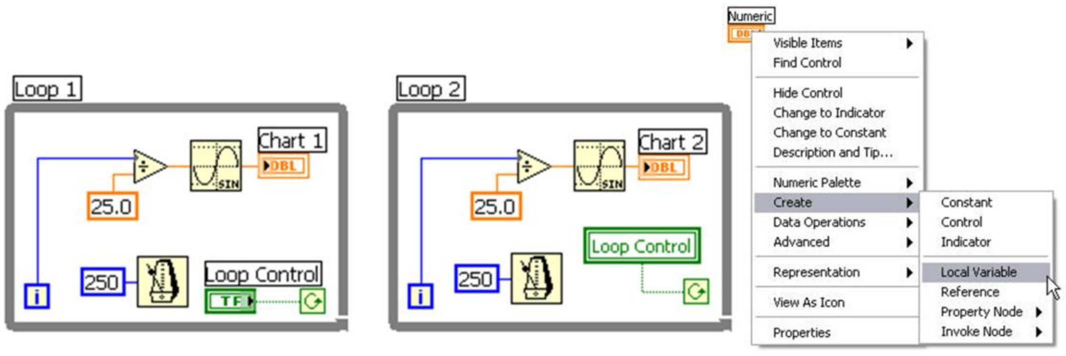
An input of 0 would result in an output of 5 for the first iteration, 10 for the second iteration, and 15 for the third iteration. Said another way, you use shift registers to retain values from one iteration to the next. They are valuable for many applications that have memory or feedback between states. The feedback node (pictured below) is another representation of the same concept. Both programs pictured behave the same.



See **Help»Search the LabVIEW Help…** for more information.

Local Variables

- Local variables allow data to be passed between parallel loops
- You can read or write a single control or indicator from more than one location in the program
  - Local variables break the dataflow paradigm and should be used sparingly



Sometimes you may need to access a front panel object from more than one place on the block diagram or to pass data between structures that you cannot connect by a wire. To accomplish these tasks, you would use a *local variable*.

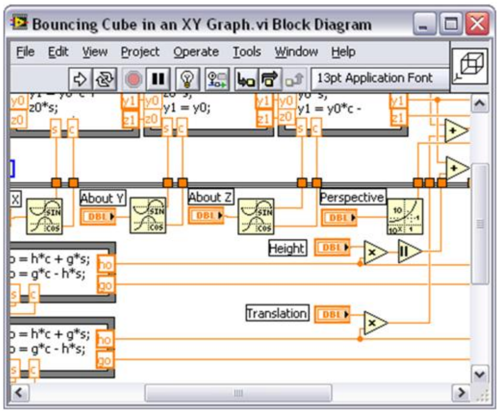
Local variables are located on the **Functions** palette under **Programming»Structures**.

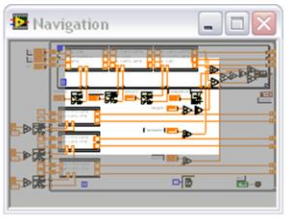
When you place a local variable on the diagram, it contains by default the name (owned label) of the first object you placed on the front panel.

You use a local variable by first selecting the object you want to access. You can either click on the local variable with the Operating tool and select the object you want to access, or right-click on the local variable and choose the object from the **Select Item** menu.

Next, you must decide to either read or write to the object. Right-click on the local variable and choose **Change to Read** or **Change to Write**.

LabVIEW Navigation Window


Bouncing Cube in an XY Graph.vi Block Diagram
File Edit View Project Operate Tools Window Help
13pt Application Font
About Y
About Z
Perspective
Height
Translation

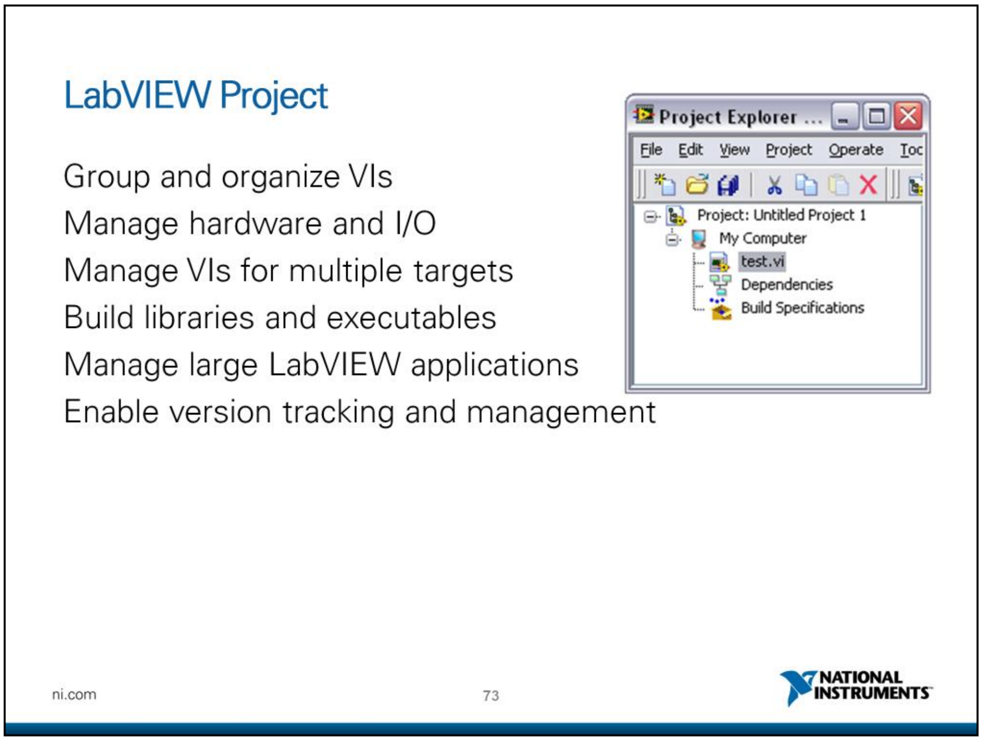


- Shows the current region of view compared to entire front panel or block diagram
- Great for large programs

Organize and reduce program visual size with subVIs

Select **View»Show Navigation Window** to display this window.

Use the window to navigate large front panels or block diagrams. Click an area of the image in the **Navigation Window** to display that area in the front panel or block diagram window. You also can click and drag the image in the **Navigation Window** to scroll through the front panel or block diagram.



LabVIEW Project

Group and organize VIs
Manage hardware and I/O
Manage VIs for multiple targets
Build libraries and executables
Manage large LabVIEW applications
Enable version tracking and management

Project Explorer ...
File Edit View Project Operate Too
Project: Untitled Project 1
My Computer
test.vi
Dependencies
Build Specifications

LabVIEW Project

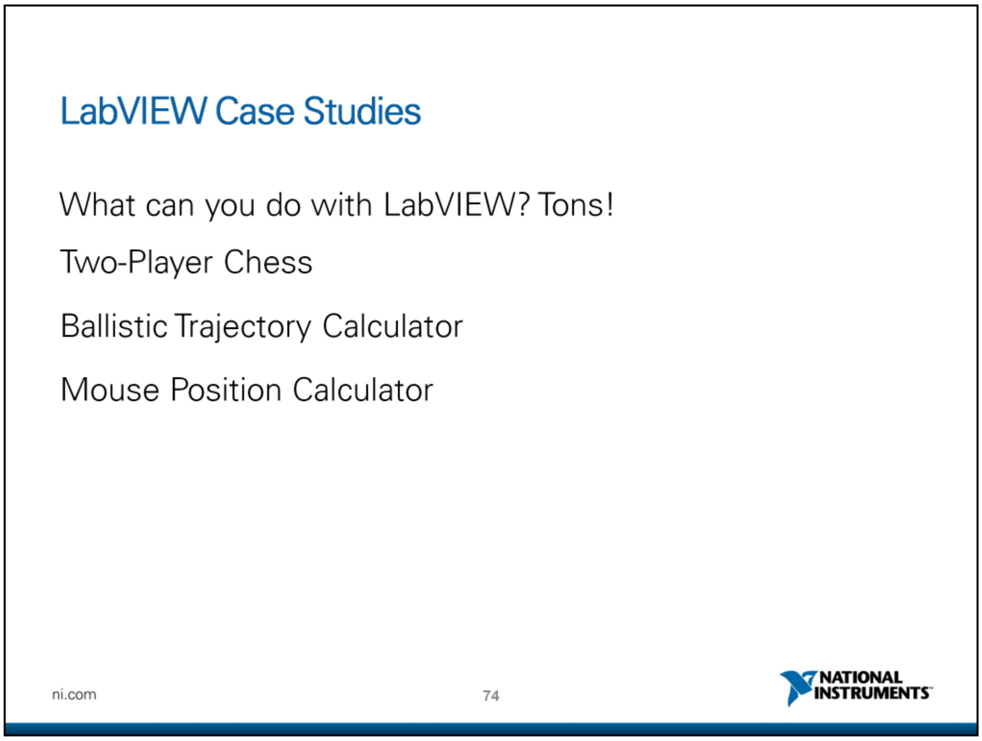
Use projects to group together LabVIEW and other files, create build specifications, and deploy or download files to targets. A target is a device or machine on which a VI runs. When you save a project, LabVIEW creates a project file (`.lvproj`), which includes configuration information, build information, deployment information, references to files in the project, and so on.

You must use a project to build stand-alone applications and shared libraries. You also must use a project to work with a real-time, FPGA, or PDA target. Refer to the specific module documentation for more information about using projects with the LabVIEW Real-Time, LabVIEW FPGA, and LabVIEW Mobile modules.

Project-style LabVIEW Plug and Play instrument drivers use the project and project library features in LabVIEW 8.0. You can use project-style drivers in the same way as previous LabVIEW Plug and Play drivers.

Project Explorer Window

Use the Project Explorer window to create and edit projects. Select **File»New Project** to display the Project Explorer window. You also can select **Project»New Project** or select **File»New** and then select **Empty Project** in the New dialog box to display the Project Explorer window.



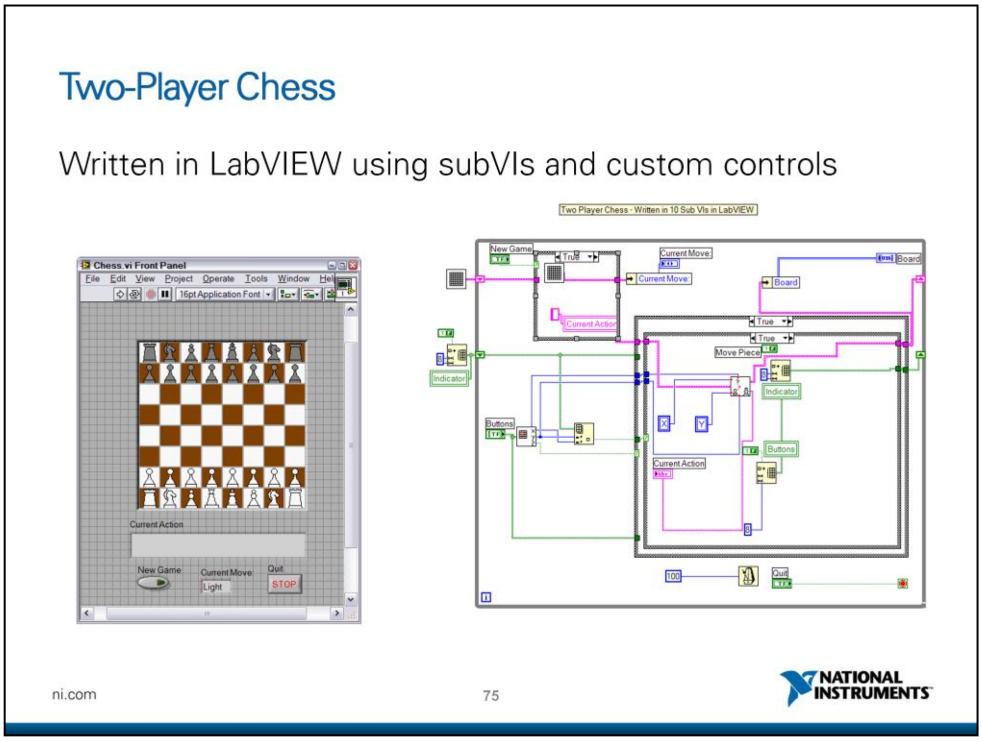
LabVIEW Case Studies

What can you do with LabVIEW? Tons!

Two-Player Chess

Ballistic Trajectory Calculator

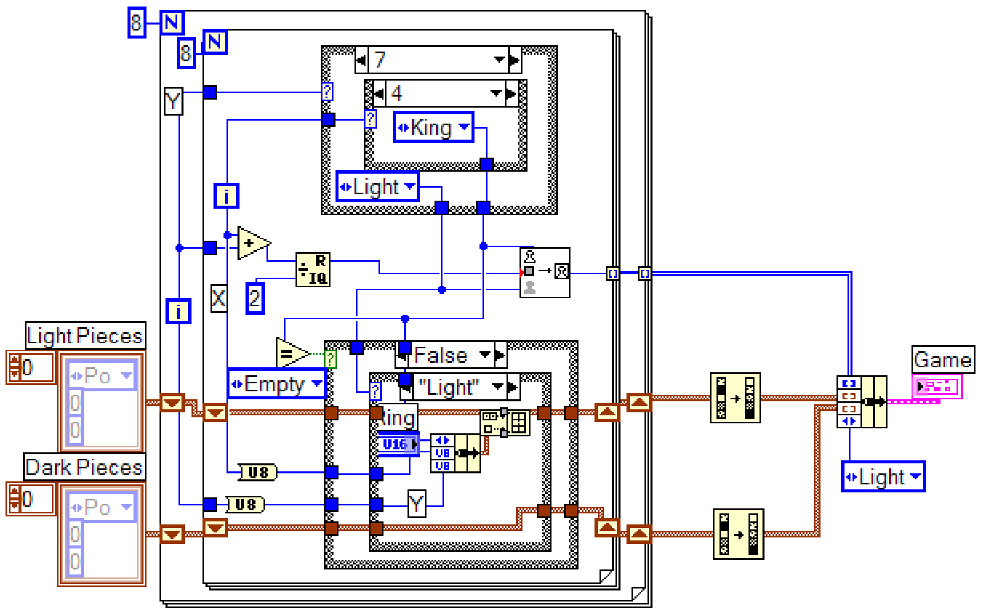
Mouse Position Calculator

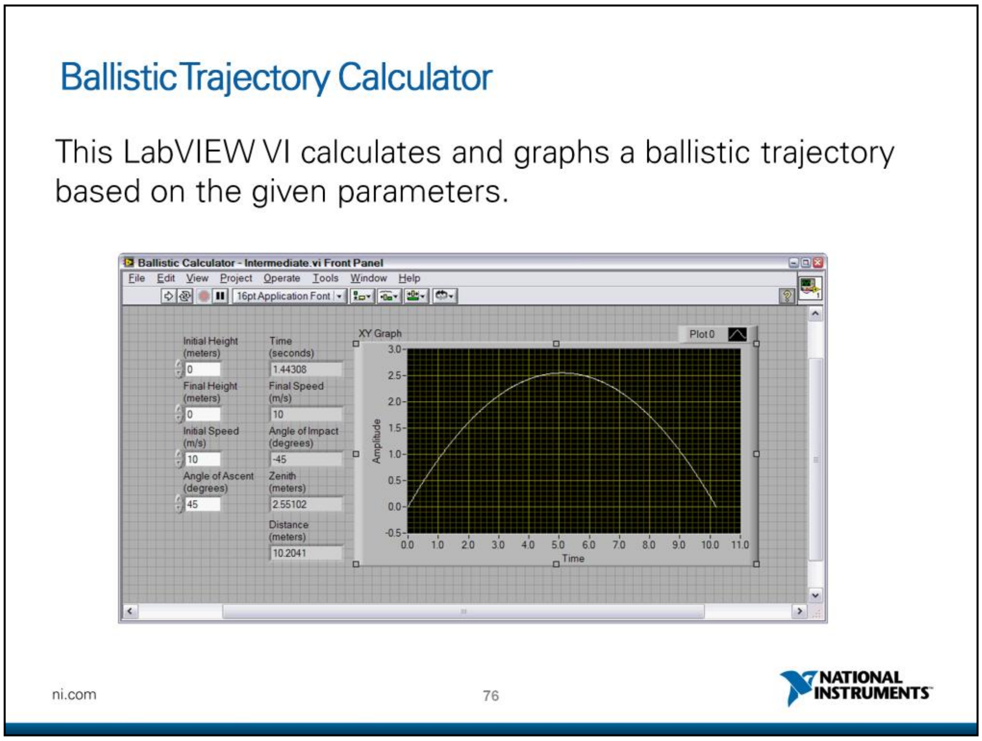


Two-Player Chess

Written in LabVIEW using subVIs and custom controls

This LabVIEW VI allows two human players to play chess. Using several custom controls and many of the LabVIEW constructs that you've worked with in this manual, the VI determines legitimate moves and displays the game in real time. The block diagram shown in the above slide handles the high-level management of the subVIs and acts as the user interface. The block diagram shown below is used to reset the board.





Ballistic Trajectory Calculator

This LabVIEW VI calculates and graphs a ballistic trajectory based on the given parameters.

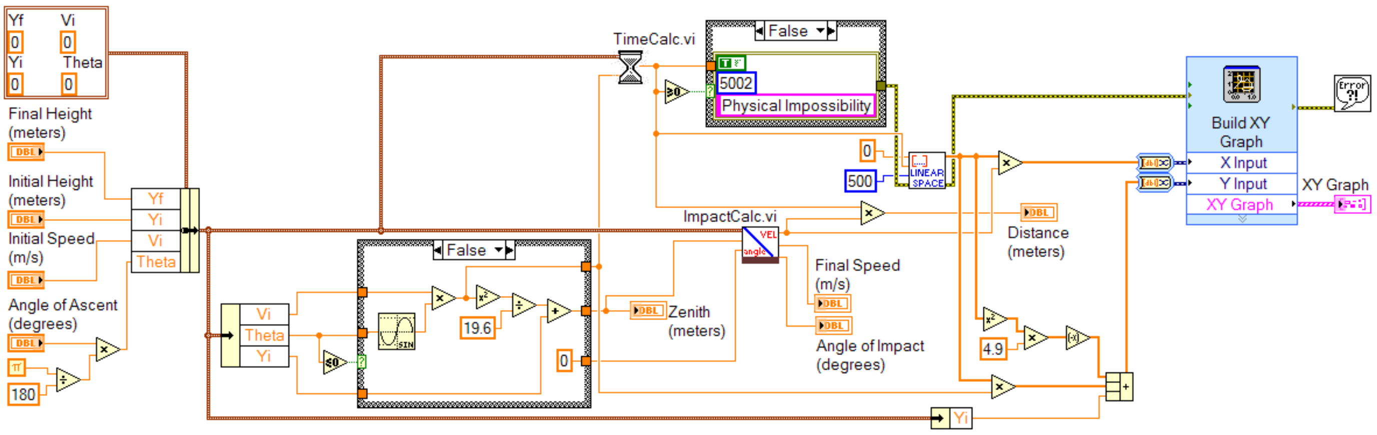
Ballistic Calculator - Intermediate.vi Front Panel

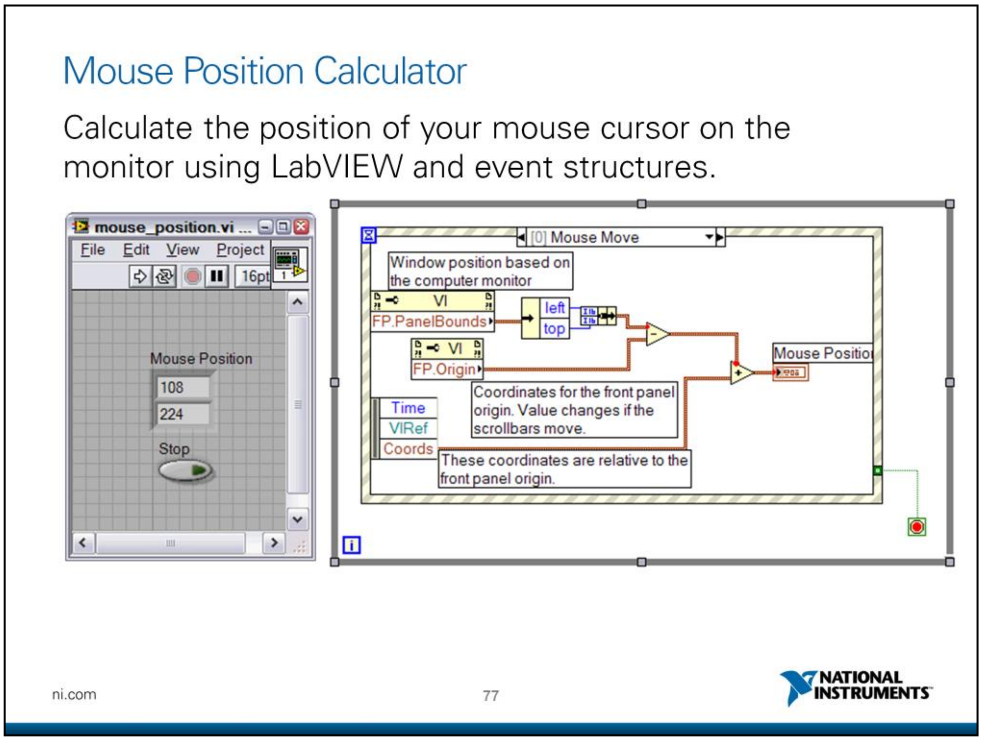
File Edit View Project Operate Tools Window Help

16pt Application Font

Initial Height (meters)	0
Final Height (meters)	0
Initial Speed (m/s)	10
Angle of Ascent (degrees)	45
Time (seconds)	1.44308
Final Speed (m/s)	10
Angle of Impact (degrees)	-45
Zenith (meters)	2.55102
Distance (meters)	10.2041

Using the code shown in the block diagram below, this LabVIEW VI calculates a projectile's ballistic trajectory based on the fundamental kinematic equations. Note the use of decision making using case structures and the modularity provided by the three subVIs.



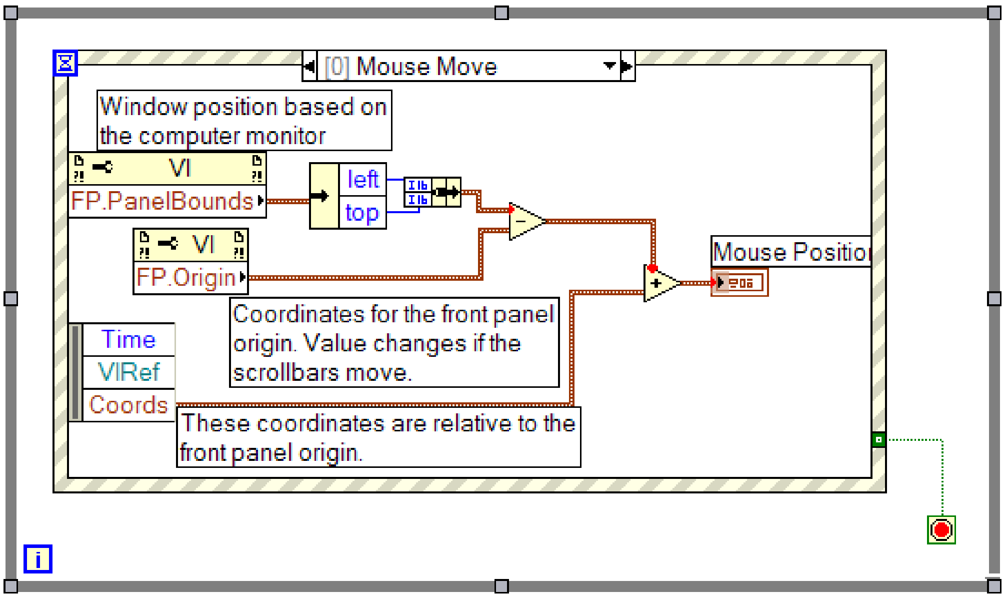


Mouse Position Calculator

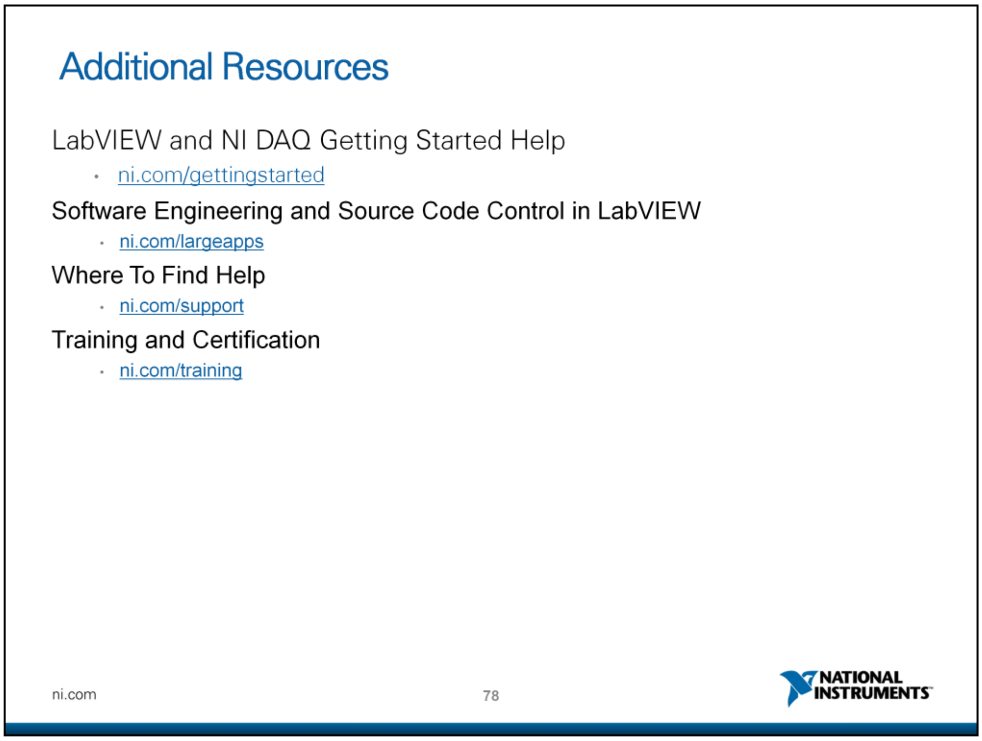
Calculate the position of your mouse cursor on the monitor using LabVIEW and event structures.

mouse_position.vi
File
Edit
View
Project
16pt
Mouse Position
108
224
Stop
[0] Mouse Move
Window position based on the computer monitor
VI
FP.PanelBounds
left
top
FP.Origin
Mouse Position
Time
VIRef
Coords
Coordinates for the front panel origin. Value changes if the scrollbars move.
These coordinates are relative to the front panel origin.

This VI uses several more advanced LabVIEW functions to calculate the position of your mouse cursor on the monitor. The VI uses two key elements. The first, event structures, are like case structures, but they are triggered from events instead of front panel elements. These events can include moving or clicking the mouse, using the keyboard, and many, many others. The other element that the VI uses are property nodes. With property nodes, you can read or set many properties of various VI elements and elements of the system. These are just some of the many powerful tools in LabVIEW that remain for you to discover.



[0] Mouse Move
Window position based on the computer monitor
VI
FP.PanelBounds
left
top
VI
FP.Origin
Mouse Position
Time
VIRef
Coords
Coordinates for the front panel origin. Value changes if the scrollbars move.
These coordinates are relative to the front panel origin.



Additional Resources

LabVIEW and NI DAQ Getting Started Help

- ni.com/gettingstarted

Software Engineering and Source Code Control in LabVIEW

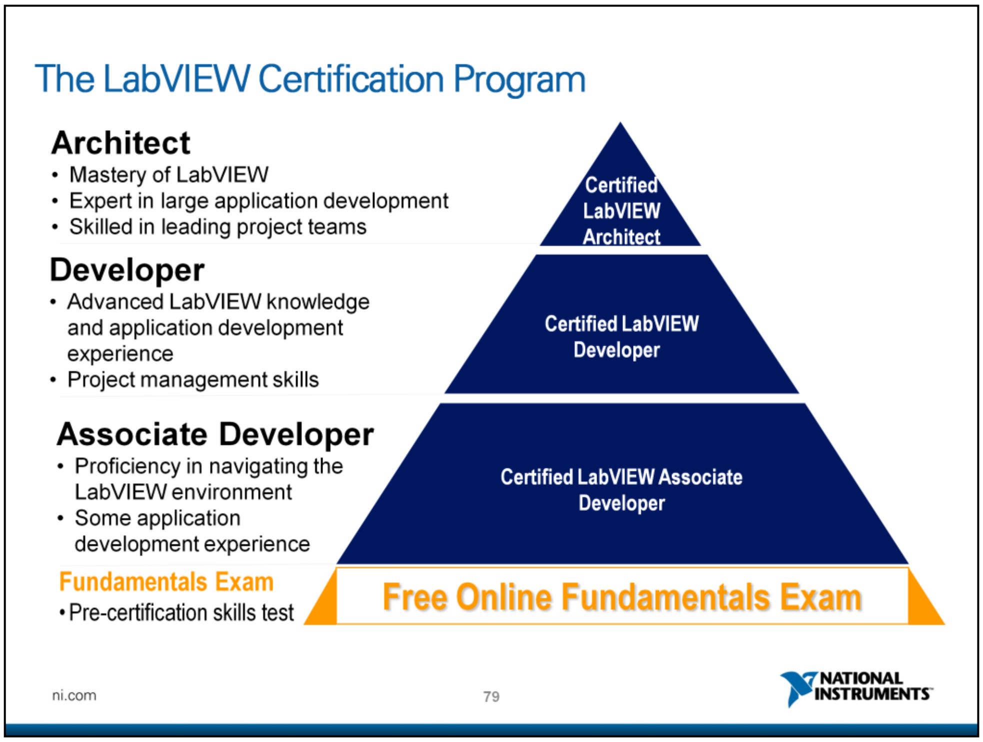
- ni.com/largeapps

Where To Find Help

- ni.com/support

Training and Certification

- ni.com/training



The LabVIEW Certification Program

Architect
- Mastery of LabVIEW
- Expert in large application development
- Skilled in leading project teams

Developer
- Advanced LabVIEW knowledge and application development experience
- Project management skills

Associate Developer
- Proficiency in navigating the LabVIEW environment
- Some application development experience

Fundamentals Exam
- Pre-certification skills test

Certified LabVIEW Architect
Certified LabVIEW Developer
Certified LabVIEW Associate Developer
Free Online Fundamentals Exam

ni.com 79 NATIONAL INSTRUMENTS

Today, more and more companies and hiring managers are looking for LabVIEW expertise when they evaluate job candidates. The LabVIEW Certification Program is built on a series of professional exams. LabVIEW certifications are used to validate LabVIEW expertise and skills for employment opportunities and for project bids.

The Certified LabVIEW Associate Developer is the first step for LabVIEW certification and it demonstrates a strong foundation in using LabVIEW and the LabVIEW environment. As a student, your Certified LabVIEW Associate Developer certification differentiates your LabVIEW skills for employment opportunities and helps potential employers recognize your LabVIEW expertise. The CLAD is a one-hour multiple-choice exam conducted at Pearson VUE testing centers around the country. The exam covers multiple topics on the LabVIEW environment including dataflow concepts, programming structures, advanced file I/O techniques, modular programming practices, VI object properties, and control references.

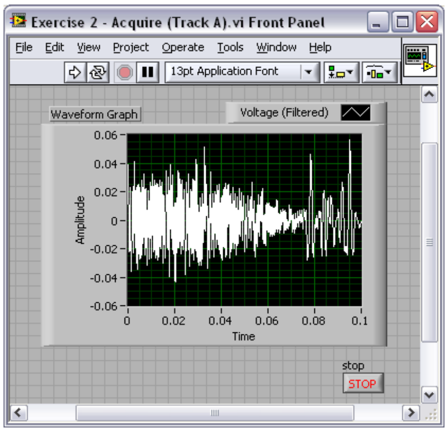
Thinking about getting your CLAD certification? Take the free online LabVIEW Fundamentals Exam as a sample test.

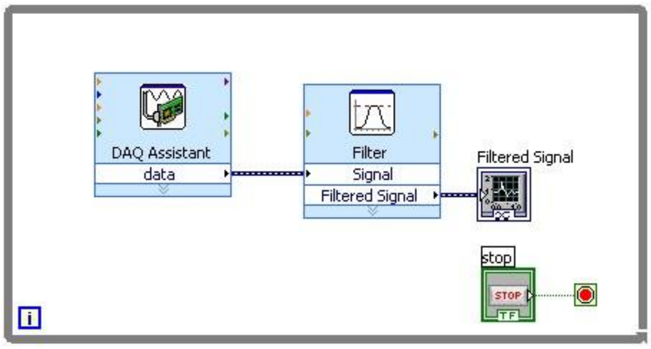
The Certified LabVIEW Developer and Architect professional certifications validate advanced LabVIEW knowledge and application development experience. Additionally, the Architect certification also demonstrates skills in leading project teams and large application development experience. These four-hour practical exams are conducted by National Instruments.

The NI LabVIEW Academy provides classroom curriculum and hands-on exercises to colleges and universities. After completion of the LabVIEW Academy program, students have the knowledge and tools to attempt the Certified LabVIEW Associate Developer certification exam with confidence.

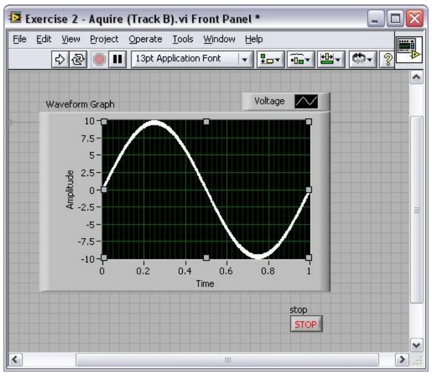
Solutions

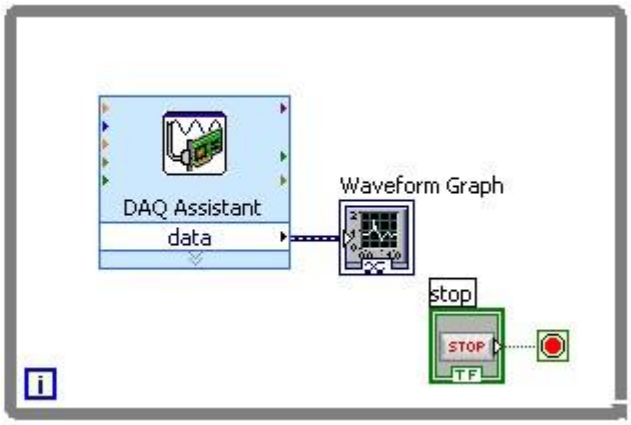
Exercise 2 Track A


Exercise 2 - Acquire (Track A).vi Front Panel
File Edit View Project Operate Tools Window Help
13pt Application Font
Waveform Graph
Voltage (Filtered)
Amplitude
Time
stop
STOP

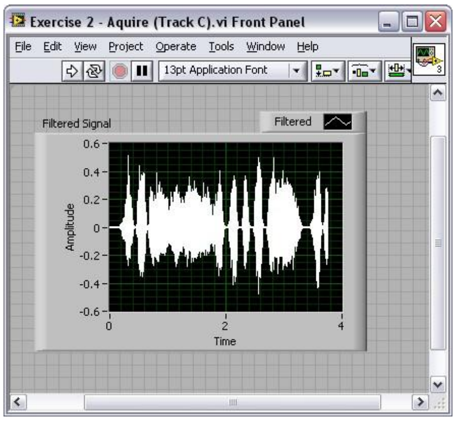


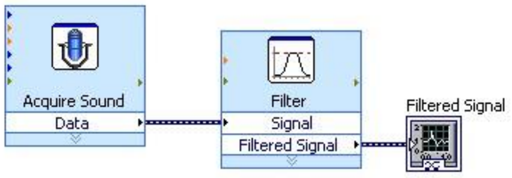
Exercise 2 Track B


Exercise 2 - Aquire (Track B).vi Front Panel *
File Edit View Project Operate Tools Window Help
13pt Application Font
Waveform Graph
Voltage
Amplitude
Time
stop
STOP

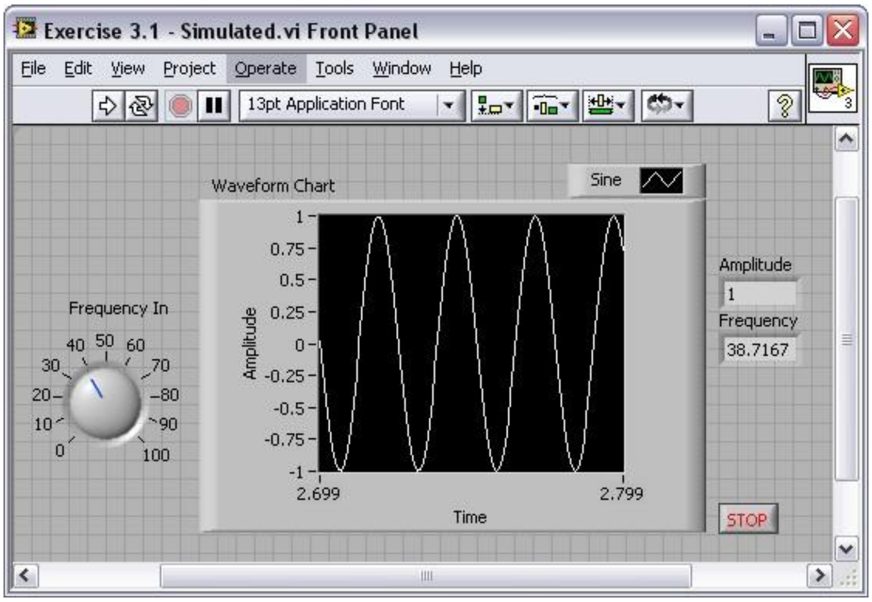


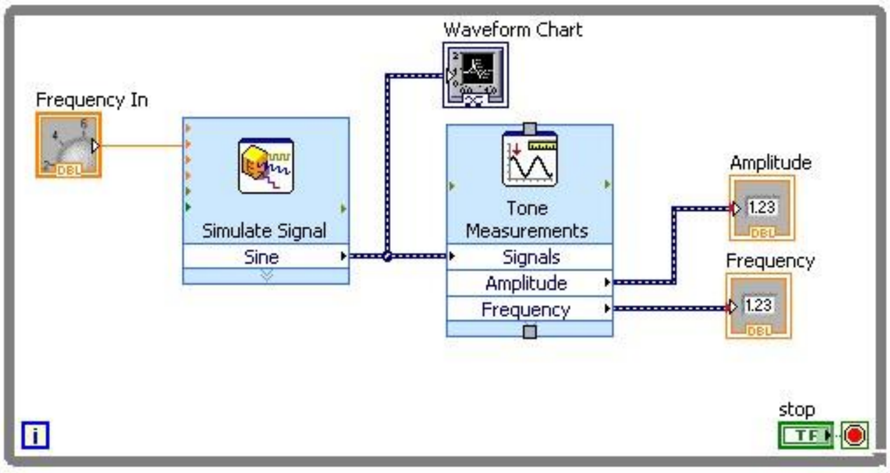
Exercise 2 Track C


Exercise 2 - Aquire (Track C).vi Front Panel
File Edit View Project Operate Tools Window Help
13pt Application Font
Filtered Signal
Filtered
Amplitude
Time

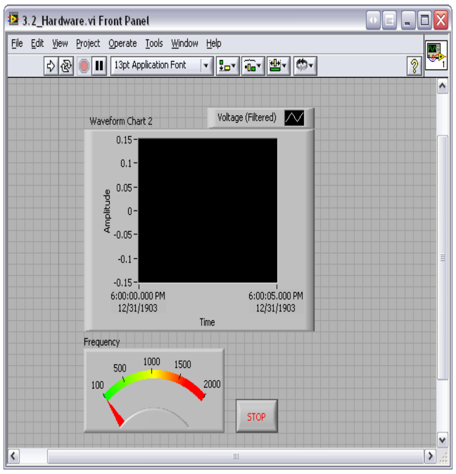


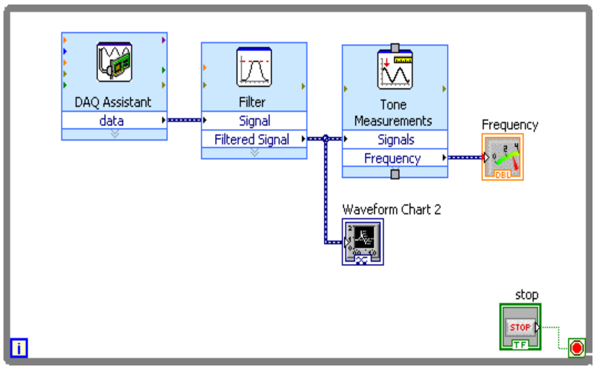
Exercise 3.1 Track A, B, and C



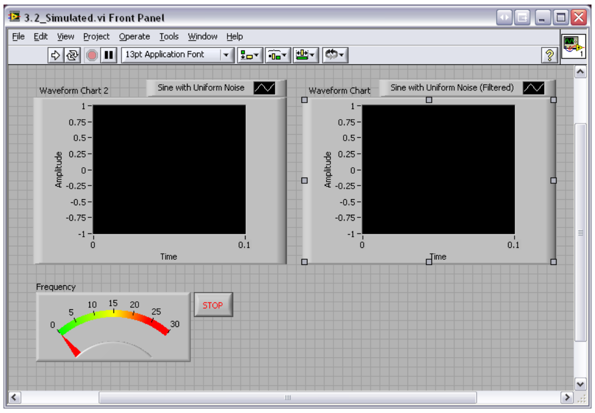


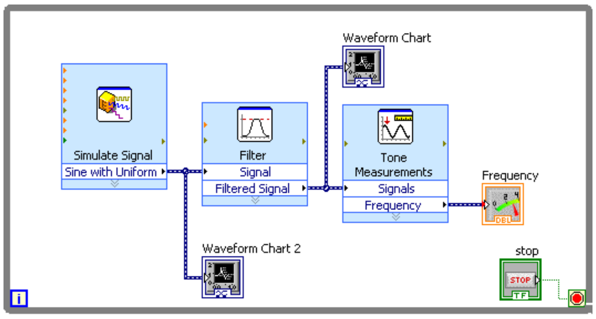
Exercise 3.2 Track A


3.2_Hardware.vi Front Panel
File Edit View Project Operate Tools Window Help
13pt Application Font
Waveform Chart 2
Voltage (Filtered)
Amplitude
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
6:00:00.000 PM 12/31/1903
6:00:05.000 PM 12/31/1903
Time
Frequency
100
500
1000
1500
2000
STOP

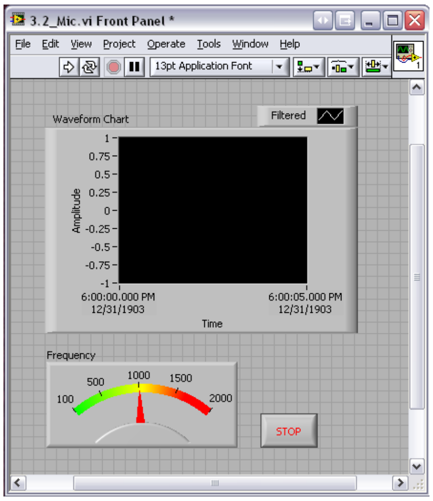


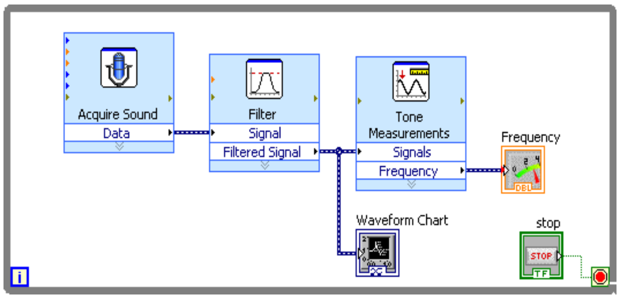
Exercise 3.2 Track B



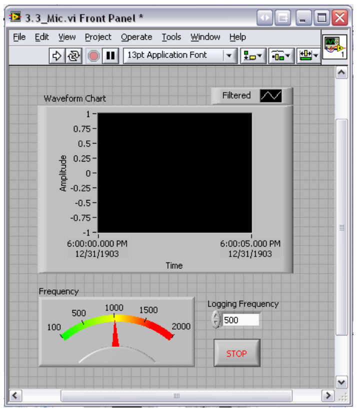


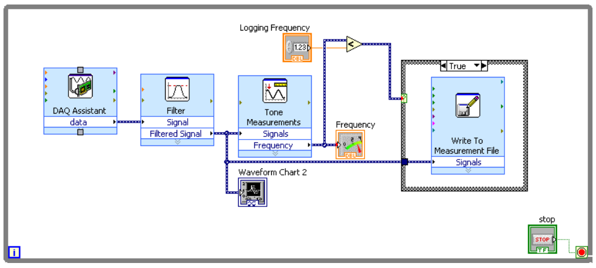
Exercise 3.2 Track C



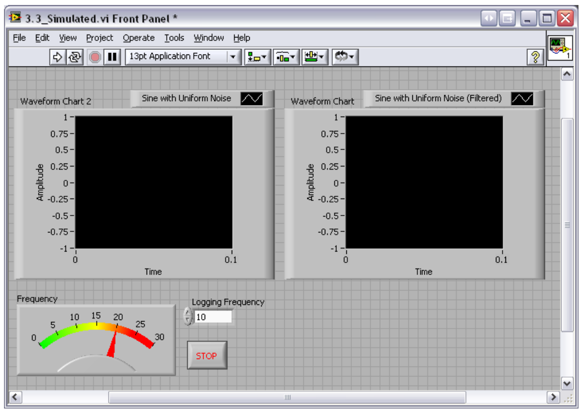


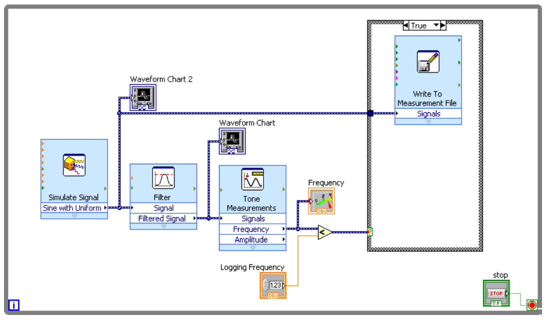
Exercise 3.2 Track A



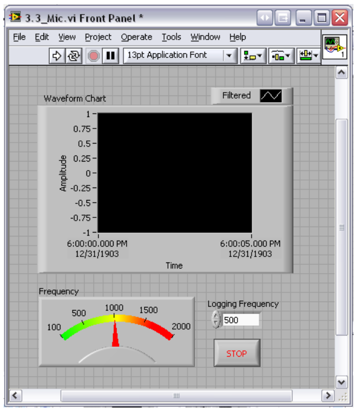


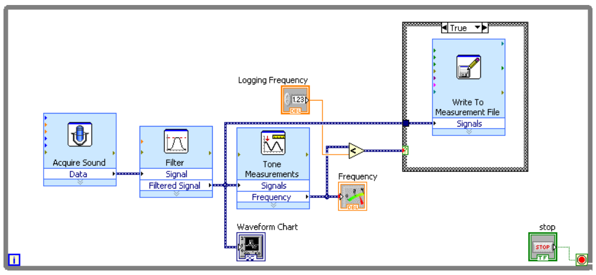
Exercise 3.2 Track B



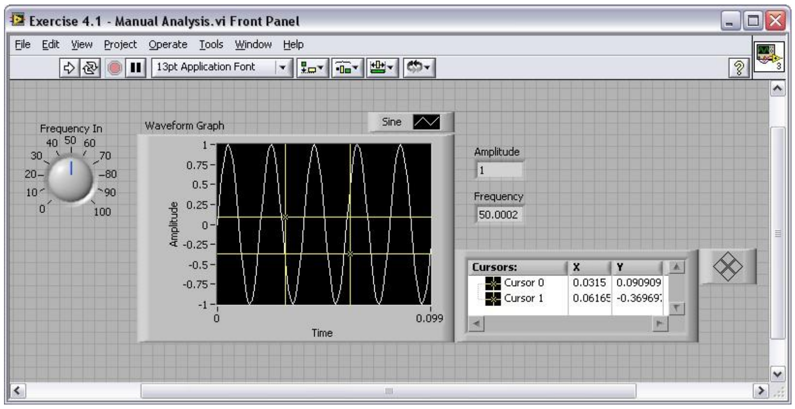


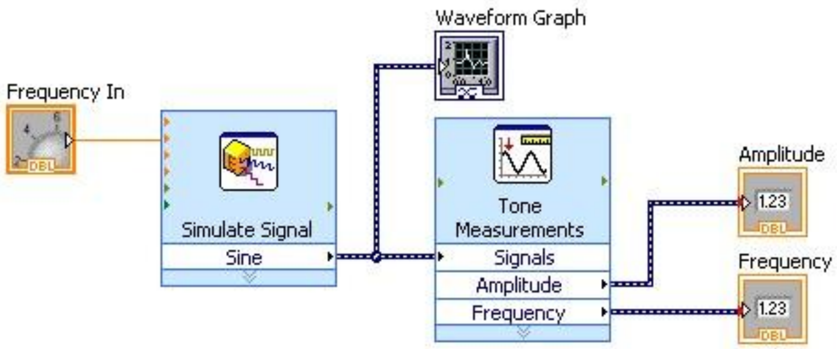
Exercise 3.2 Track C



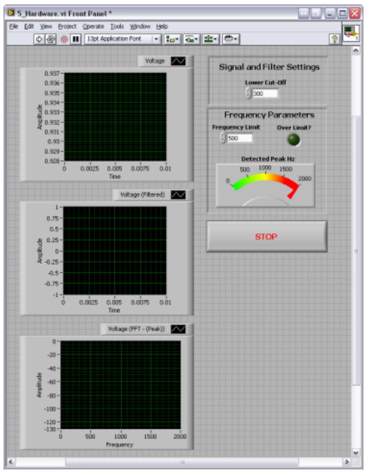


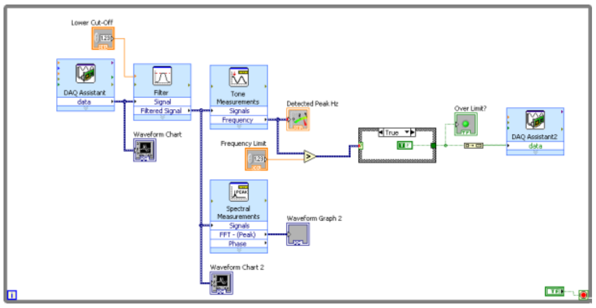
Exercise 4 Track A, B, and C



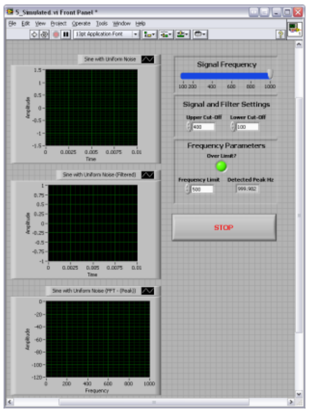


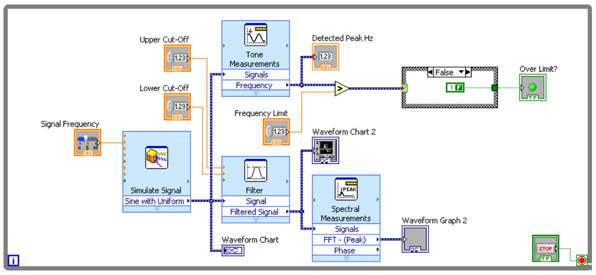
Exercise 5 Track A



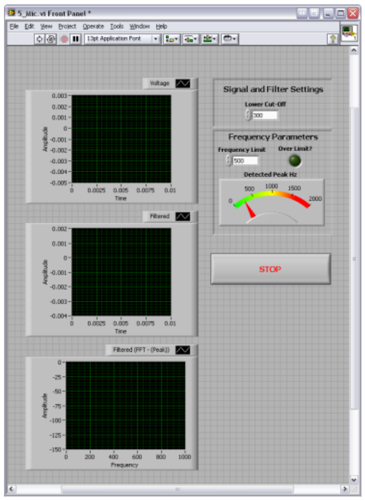


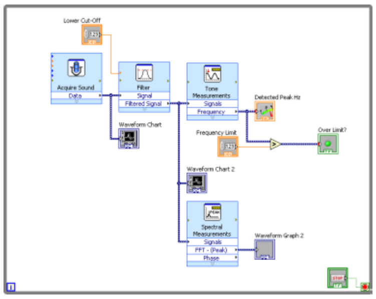
Exercise 5 Track B





Exercise 5 Track C


5_Mic.vi Front Panel *
Voltage
Filtered
Filtered (FFT - (Peak))
Signal and Filter Settings
Lower Cut-Off
Frequency Parameters
Frequency Limit
Over Limit?
Detected Peak Hz
STOP


Lower Cut-Off
Acquire Sound
Filter
Tone Measurements
Detected Peak Hz
Over Limit?
Waveform Chart
Frequency Limit
Waveform Chart 2
Spectral Measurements
Waveform Graph 2